Bench-MR: A Motion Planning Benchmark for Wheeled Mobile Robots

Eric Heiden*[1], Luigi Palmieri*[2], Leonard Bruns[3], Kai O. Arras[2], Gaurav S. Sukhatme[1] and Sven Koenig[1]

***Abstract*—Planning smooth and energy-efficient paths for wheeled mobile robots is a central task for applications ranging from autonomous driving to service and intralogistic robotics. Over the past decades, several sampling-based motion-planning algorithms, extend functions and post-smoothing algorithms have been introduced for such motion-planning systems. Choosing the best combination of components for an application is a tedious exercise, even for expert users. We therefore present Bench-MR, the first open-source motion-planning benchmarking framework designed for sampling-based motion planning for nonholonomic, wheeled mobile robots. Unlike related software suites, Bench-MR is an easy-to-use and comprehensive benchmarking framework that provides a large variety of sampling-based motion-planning algorithms, extend functions, collision checkers, post-smoothing algorithms and optimization criteria. It aids practitioners and researchers in designing, testing, and evaluating motion-planning systems, and comparing them against the state of the art on complex navigation scenarios through many performance metrics. Through several experiments, we demonstrate how Bench-MR can be used to gain extensive insights from the benchmarking results it generates.**

***Index Terms*—Nonholonomic Motion Planning; Wheeled Robots; Software Tools for Benchmarking and Reproducibility.**

I. Introduction

MOTION PLANNING is a central component for autonomous navigation in various real-world domains, such as autonomous driving, warehouse logistics and service robotics [1]. Over the years, many different sampling-based motion-planning algorithms and related components, such as extend functions and post-smoothing algorithms, have been introduced for such motion-planning systems. Choosing from this plethora of components to create a motion-planning system or to design a novel component for one is a complex task that requires significant effort in testing. To reduce this effort, we have created **Bench-MR**, the first open-source benchmarking framework designed for sampling-based motion planning for nonholonomic, wheeled mobile robots in complex navigation scenarios resembling real-world applications.

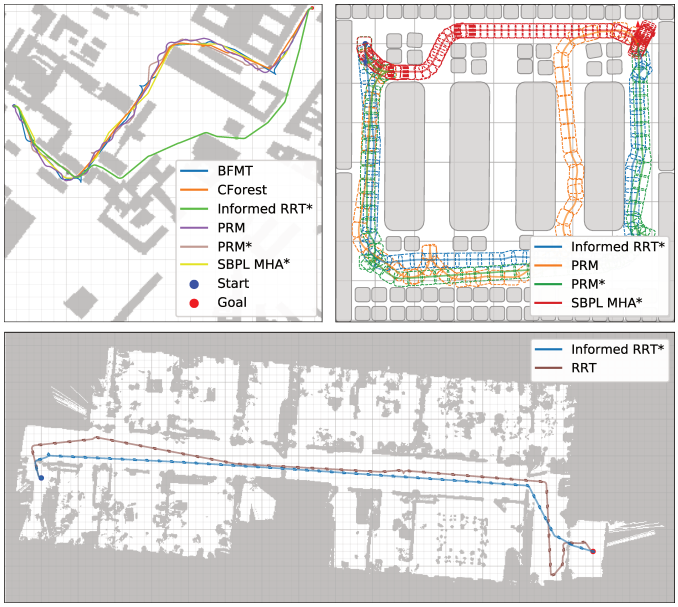


Fig. 1. Selection of environments provided by Bench-MR: City grid from the Moving AI path-finding benchmark [2] (top left), polygon-based warehouse environment (top right), and thresholded occupancy grid from the Freiburg SLAM dataset [3] (bottom).

Bench-MR is based on two main pillars, namely the motion-planning components (consisting of the sampling-based motion planning algorithms, extend functions, collision checkers, post-smoothing algorithms and optimization criteria) and the evaluation components (consisting of the navigation scenarios and performance metrics), see Fig. 2. We chose all these components carefully to match the application constraints. For example, we focus on polygon-based collision checking since it presents a challenge for motion-planning algorithms which make inefficient use of collision checking. Furthermore, we support the evaluation of motion-planning systems for particular settings of navigation scenarios, such as varying obstacle density. Overall, Bench-MR is a highly configurable and expandable software suite with representative state-of-the-art motion-planning and evaluation components. It helps one to gain novel insights, such as *i)* how some combinations of motion-planning and post-smoothing algorithms achieve better performance than asymptotically (near) optimal motion-planning algorithms or *ii)* how changes of the obstacle density in navigation scenarios can affect the planning efficiency and the resulting path quality.

Much of Bench-MR builds on the Open Motion Planning Library (OMPL) [4], but we also provide interfaces to implementations of motion-planning algorithms (such as SBPL

* Equal contribution

Manuscript received: October 15, 2020; Revised January 18, 2021; Accepted February 14, 2021.

This paper was recommended for publication by Associate Editor N. Amato upon evaluation of the Reviewers' comments.

[1]E. Heiden, G. S. Sukhatme and S. Koenig are with the Department of Computer Science, University of Southern California, Los Angeles, USA {heiden, gaurav, skoenig}@usc.edu.

[2]L. Palmieri and K. O. Arras are with Robert Bosch GmbH, Corporate Research, Stuttgart, Germany {Luigi.Palmieri, KaiOliver.Arras}@de.bosch.com.

[3]L. Bruns is with the Division of Robotics, Perception and Learning (RPL), KTH Royal Institute of Technology, Stockholm, Sweden leonardb@kth.se.

Digital Object Identifier (DOI): see top of this page.

planners [5]) and extend functions (such as POSQ [6] and continuous-curvature steering [7]) outside of OMPL. Thus, Bench-MR offers users access to state-of-the-art components of sampling-based motion-planning systems for wheeled mobile robots, while being less confined to particular implementations of these components.

II. Related Work

Several researchers have recently introduced benchmarking frameworks for analyzing motion-planning algorithms for different robotic systems. We discuss some of the most prominent ones in the following.

Sturtevant [2] has introduced a benchmarking framework for path-planning algorithms for robotic systems without kinematic constraints. The Moving AI path-finding benchmark provides many navigation scenarios on different grid-based environments, such as city grids. Bench-MR includes some of their environments (and supports their format) but additionally it provides many other environment classes, motion-planning components and evaluation components for wheeled mobile robots.

Luo et al. [8] have introduced a benchmarking framework for asymptotically optimal motion-planning that supports only straight-line connections and compares them only on four navigation scenarios. Bench-MR, on the other hand, provides many diverse navigation scenarios for wheeled mobile robots.

Moll et al. [9] have introduced a general benchmarking framework for motion-planning algorithms that is highly coupled with OMPL. It is highly customizable but lacks of specific navigation scenarios for wheeled mobile robots. Bench-MR, on the other hand, provides navigation scenarios, performance metrics and extend functions for wheeled mobile robots and, similar to Cohen et al. [10], different classes of motion-planning algorithms, including lattice-based planners.

Althoff et al. [11] have introduced a benchmarking framework for autonomous cars driving on roads. Bench-MR, on the other hand, focuses on wheeled mobile robots in complex and cluttered static (indoor and outdoor) environments.

Additionally the website [12] provides several benchmarks for different robotic systems but contains only a small number of navigation scenarios for wheeled mobile robots. Instead PathBench [13] is a framework for testing recent machine learning based algorithms for planning in 2D or 3D grid environments without focusing on mobile robots.

A number of authors [14], [15], [16], [17] have introduced benchmarking frameworks for motion-planning algorithms in dynamic environments. Bench-MR, on the other hand, focuses on motion planning in static environments, which is a fundamental operation often performed during robot navigation in dynamic environments.

III. Architecture of Bench-MR

Bench-MR is split into a Python front-end and a C++ back-end, see Fig. 2. The front-end provides a flexible interface for setting up and performing evaluations of motion-planning systems through Jupyter notebooks. For example, the front-end allows the user to select appropriate navigation scenarios (such as environment classes) and performance metrics related to the planning efficiency and the resulting motion quality. It then provides the user with extensive evaluation reports and plotting capabilities. The back-end performs the (compute-intensive) evaluations by using the motion-planning components in the blue box and the evaluation components in the orange box. We chose all components based on their scientific impact and their popularity in the open-source community [4], [18], [7]. Our choices are presented in Sec. IV-V. JSON files are used for communicating both settings from the front-end to the back-end and the evaluation results in the opposite direction. The open-source code of Bench-MR is available at https://github.com/robot-motion/bench-mr. This website also contains extensive documentation, including tutorials and examples, and up-to-date benchmarking results, that are automatically generated.

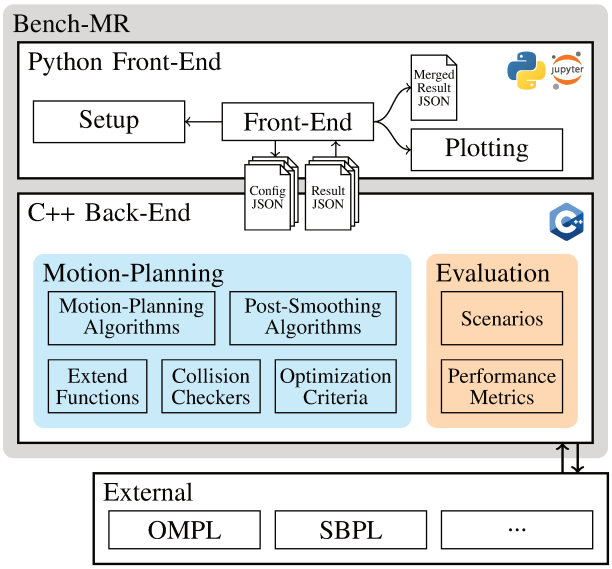


Fig. 2. Architecture of Bench-MR. The components necessary for motion planning are shown in the box on the left (turquoise), and the utilities used in the evaluation are shown in the box on the right (orange). The implementation is split into a C++ back-end for running the performance-critical motion-planning components, and a Python front-end for providing a flexible interface to the design and evaluation of the benchmark scenarios through Jupyter notebooks.

Bench-MR provides interfaces to two existing open-source motion-planning libraries, namely OMPL [4] and SBPL [5], enabling the user to utilize their components as part of Bench-MR. We expose many settings from OMPL and SBPL through the Python interface, to allow the user to change the parameters of their components. Cross-component settings in Bench-MR (such as the computation time limit) can be changed via a common interface.

IV. Bench-MR Planning Components

In this section, we explain the Bench-MR motion-planning components.

A. Sampling-Based Motion-Planning Algorithms

Bench-MR provides many different sampling-based motion-planning algorithms that belong to to three different classes

(as suggested by prior work, such as [19], [20], [21]): *feasible planners*, *asymptotically (near) optimal planners* and *lattice-based planners*.[1] For feasible and asymptotically (near) optimal planners, Bench-MR provides the option to use random sampling with a uniform distribution and goal biasing or deterministic Halton sampling [21], [19], [22]. We choose the most prominent open-source implementation for each class.

1) Feasible Planners: Feasible planners eventually find a path with probability one but not necessarily an optimal path. Bench-MR currently provides feasible planners from OMPL (such as RRT [23], PRM [24], SPARS [25], RRT [23], [26] using random forward propagation, EST [27], SBL [28] and STRIDE [29]).

2) Asymptotically (Near) Optimal Planners: Asymptotically (near) optimal planners eventually find an optimal path with probability one. Bench-MR currently provides optimization-based planners from OMPL (such as RRT* and PRM* [30], BFMT [31], RRT# [32]), informed search-based planners (such as Informed RRT* [33], SORRT* [34] and BIT* [35]), CForest [36] and near-optimal planners (such as SST [37], an asymptotically near-optimal incremental version of RRT, SPARS [25] and SPARS2 [38]).

3) Lattice-Based Planners: Lattice-based planners use state lattices with predefined motion primitives that encode differential constraints [39]. Bench-MR currently provides lattice-based planners from SBPL (such as ARA* [5], AD* [18], MHA* [40] and ANA* [41]).

B. Extend Functions

Depending on the class of a sampling-based motion-planning algorithm, Bench-MR provides two classes of extend functions, namely those that use random forward propagation for a given robot dynamical model and those that solve a two-point boundary value problem [42] to connect two given robot configurations exactly for a given steer function. We refer the reader to [26] for an analysis of the properties of both classes. We also include the predefined motion primitives for lattice-based planners here since they can be understood as a discrete set of predefined controls.

1) Robot Dynamics Models: Bench-MR provides two robot dynamics models, namely a kinematic car ($\dot{x} = v\cos\theta, \dot{y} = v\sin\theta, \dot{\theta} = {}^{v}/_{L} \cdot \tan\delta$) and a kinematic single-track model ($\dot{x} = v\cos\theta, \dot{y} = v\sin\theta, \dot{\theta} = {}^{v}/_{L} \cdot \tan\delta, \dot{\delta} = v_{\delta}$), where x and y are the Cartesian coordinates according to a fixed world frame, L is the length of the car, v is the tangential velocity, θ is the heading, δ is the steering angle and $\dot{\delta}$ is its rate [1].

2) Steer Functions: Bench-MR provides common steer functions, namely Dubins [43], Reeds-Shepp [44], Continuous Curvature [45], [7] and POSQ [6], [46].

3) Motion Primitives: Bench-MR provides motion primitives from SBPL but also supports changing them by means of the primitive file interface of SBPL.

[1]For the sake of brevity, we do not list all included planners with detailed explanations and instead direct the reader to the corresponding references.

C. Collision Checkers

Bench-MR provides a two-dimensional grid-based approach to collision checking, which checks whether the robot (modeled as a polygon or single point) collides with blocked cells. It also includes a two-dimensional polygon-based approach to collision checking, which uses the *separating axis theorem* [47] to check whether the robot (modeled as a convex polygon) intersects with obstacles (also modeled as convex polygons). Finally, Bench-MR provides the distance field, represented as a grid whose cells are annotated with the distance to the closest obstacle, for all environment classes.

D. Post-Smoothing Algorithms

Bench-MR includes several post-smoothing algorithms from OMPL, such as B-Spline, Shortcut and SimplifyMax [4]. It also includes the recently introduced GRradient-Informed Post Smoothing (GRIPS) algorithm [48], a hybrid approach that combines short-cutting with locally optimized waypoint placement based on the distance field of the environment.

E. Optimization Criteria

Bench-MR provides optimization criteria by allowing user-defined cost functions for several motion-planning algorithms.

V. Bench-MR Evaluation Components

In this section, we explain the Bench-MR evaluation components.

A. Navigation Scenarios

A navigation scenario consists of a specification of the shapes of obstacles in an environment, the shape of a robot, and its start and goal poses. Bench-MR provides the two common environment classes used by motion-planning systems, namely grid-based and (convex) polygon-based environments. It provides both predefined and procedurally-generated environments for both classes.

1) Predefined Grid-Based Environments: Bench-MR provides two classes of predefined grid-based environments. It includes a selection of city grids from the Moving AI path-finding benchmark [2], consisting of city layouts of sizes ranging from 256×256 to 1024×1024 cells. An example is the `Berlin_0_256` grid in Fig. 1 (top left). It also provides image-based grids that can be created via an interface from grey-scale images by thresholding with a user-defined threshold (a common representation for maps generated by SLAM algorithms [3]). Examples are shown in Fig. 1 (bottom) and Fig. 3.

2) Procedurally-Generated Grid-Based Environments: Bench-MR provides two classes of procedurally-generated grid-based environments to allow the user to vary environment characteristics (such as the environment complexity) in small steps. It provides random outdoor-like environments (with occasional small obstacles, such as trees) with a desired percentage of blocked cells γ. These environments are generated by starting with only unblocked cells and repeatedly sampling a cell with a uniform distribution and making it

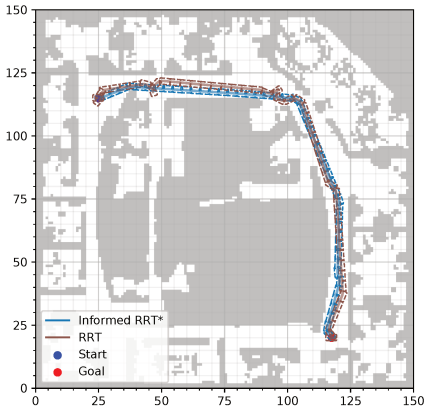


Fig. 3. Predefined grid-based environment obtained from a gray-scale image of an Intel office building [3].

blocked. Examples are shown in Fig. 4 (top). It also provides random indoor-like environments (with complex networks of rectangular spaces, such as rooms and corridors) with a desired minimum corridor width r. They are generated by starting with only blocked cells and, for a predefined number of steps, repeatedly sampling a cell with a uniform distribution and applying a modified RRT exploration algorithm to connect it to the nearest tree node with either horizontal or vertical unblocked corridors of the desired minimum corridor width. Examples are shown in Fig. 4 (bottom).

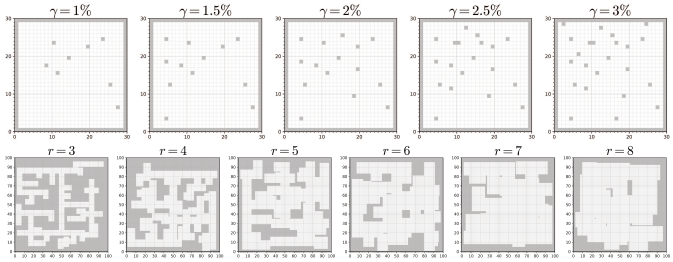


Fig. 4. Procedurally-generated grid-based environments, namely random outdoor-like environments with different percentages of blocked cells (top), and random indoor-like environments with different minimum corridor widths (bottom).

3) Predefined Polygon-Based Environments: Bench-MR provides five classes of predefined polygon-based environments, as shown in the left-most five subfigures of Fig. 5. It provides three parking scenarios in street environments where a car-like vehicle has to park between other cars, namely by *i)* pulling forward into a parking space, *iii)* backing into a parking space, and *ii)* parallel parking. Bench-MR also provides two navigation scenarios in warehouse environments where a square-shaped robot has to navigate among shelves of various sizes and irregular orientations. Additional polygon-based environments can be loaded from SVG files.

4) Procedurally-Generated Polygon-Based Environments: Bench-MR allows the user to generate their own polygon-based environments procedurally by placing (convex) polygonal obstacles into the environment. An example resembling an asteroid field is shown in the right-most subfigure of Fig. 5.

B. Performance Metrics

Bench-MR provides commonly used performance metrics for evaluating motion-planning systems with respect to their planning efficiency and resulting path quality.

1) *The success statistics* measure the percentage of found, collision-free and exact paths. Whether a path is collision-free is checked with a given collision checker. The ratio of exact paths is included since some motion-planning systems report approximate paths.
2) *The path length* measures the length in meters (m) of a path in the workspace.
3) *The maximum curvature* ($\kappa_{\max}$), *normalized curvature* (κ_{norm}) and *angle-over-length* (*AOL*) measure the smoothness of a path. Smoother paths result in less control effort and energy to steer a robot and more comfort for the passengers. Since the maximum curvature is not well-defined in the presence of cusps, we also use the normalized curvature (which is the path-length-weighted curvature along the path segments between the cusps), defined as
$$\kappa_{\text{norm}} = \sum_i \int_{\sigma_i} \kappa(\dot{\sigma}_i(t)) ||\dot{p}_{\sigma_i}(t)||_2 \, dt, \tag{1}$$
where σ_i are the path segments of path σ between the cusps, $\kappa(\dot{\sigma}(t))$ is the curvature at point $\sigma(t)$ of the path and p_σ are the x and y components of σ. Since the normalized curvature ignores cusps, we also use the *angle-over-length* (*AOL*) as a combined metric that divides the total heading change by the path length. The total heading change is computed numerically by summing the absolute angular difference between neighboring tangent vectors along the path. Following this convention, the heading change for each cusp is approximately π.
4) *The computation times* measure the time in seconds (s) required for collision checking, for extend function evaluation (namely forward integration when using forward propagation or solving the two-point boundary value problems when using steer functions), and for finding an initial path.
5) *The mean clearing distance* measures how close a path is to obstacles (reported in meters, m).
6) *The number of cusps* [45] measures how often a robot has to stop on a path and turns its wheels to abruptly change its heading.

VI. Experiments with Bench-MR

The large variety of tools provided by Bench-MR allows the user to compare various motion-planning systems on complex navigation scenarios with many performance metrics and perform ablation studies, which is a key contribution of Bench-MR that was often missing in prior work. We describe several experiments performed with Bench-MR and their results to provide examples of its use. These experiments are available as Jupyter notebooks to help the user with developing their own experiments. Our experiments with different post-smoothing algorithms and different optimization criteria resulted in novel scientific insights into the performance of sampling-based motion-planning systems.

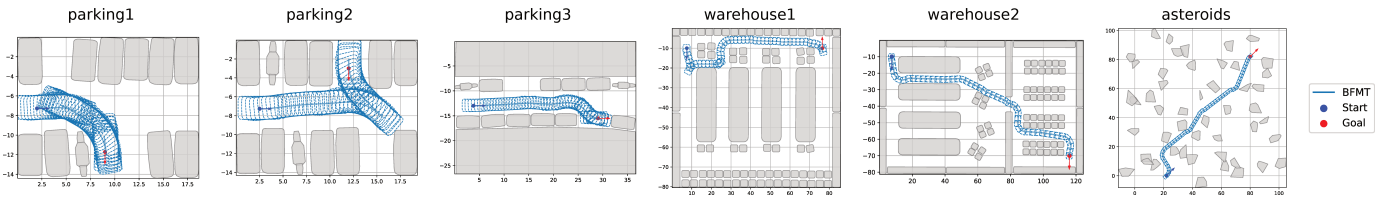


Fig. 5. Paths for polygon-based environments computed by the *Bidirectional Asymptotically Optimal Fast Marching Tree* (BFMT) motion-planning algorithm using the Reeds-Shepp steer function. The first five environments are predefined, and the right-most environment is procedurally generated.

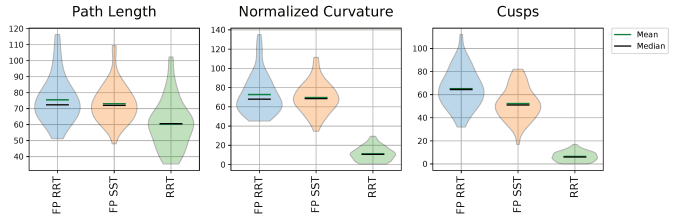


Fig. 6. Path length, normalized curvature and number of cusps of different combinations of sampling-based motion-planning algorithms and extend functions, namely RRT and SST using random forward propagation for the kinematic car model (left and center) and RRT using the Reeds-Shepp steer function (right). All performance metrics are reported with a computation time limit of 30 seconds each in 50 random indoor-like grid-based environments with a desired minimum corridor width of 5 cells.

A. Different Extend Functions

Bench-MR allows us to compare different combinations of sampling-based motion-planning algorithms and extend functions, which is important since it is often overlooked that the performance of sampling-based motion-planning algorithms depends on their extend functions [26], [6]. As an example, we compare RRT using random forward propagation for the kinematic car model, SST using random forward propagation for the kinematic car model and RRT using the Reeds-Shepp steer function. Fig. 6 shows that RRT with the Reeds-Shepp steer function achieves smaller path length, normalized curvature and number of cusps.

B. Different Post-Smoothing Algorithms

Bench-MR allows us to compare different combinations of feasible motion-planning algorithms (that find initial paths quickly) and post-smoothing algorithms (that improve the quality of the initial paths), which is important since such combinations have rarely been thoroughly evaluated [35], [37]. As an example, we compare the feasible motion-planning algorithms RRT, EST, SBL and STRIDE using the post-smoothing algorithms GRIPS, B-Spline, Shortcut and SimplifyMax against the asymptotically (near) optimal motion-planning algorithms RRT*, Informed RRT*, SORRT*, PRM*, CForest, BIT* and SPARS. The comparison is performed adopting the Reeds-Shepp extend function. Fig. 7 shows that feasible motion-planning with post-smoothing can indeed outperform asymptotically (near) optimal motion-planning algorithms in both planning efficiency and the resulting path quality. For example, RRT using the post-smoothing algorithm SimplifyMax achieves a smaller path length and about the same normalized curvature after less than one second than Informed RRT* after 60 seconds. Fig. 8 shows that the post-smoothing algorithms GRIPS and SimplifyMax often significantly decrease the path length and maximum curvature, with SimplifyMax typically running faster. The post-smoothing algorithm B-spline does not always improve the path quality, which might be due to the issue that B-splines do not translate well to paths that can be followed by the Reeds-Shepp and other steer functions, resulting in slight turns that increase the curvature.

C. Different Sampling Strategies

Bench-MR allows us to compare different sampling strategies, for example using random sampling and de-randomized approaches, such as using deterministic sampling or state lattices. As an example, we compare PRM* using random uniform sampling against PRM* using deterministic Halton sampling (both using the Reeds-Shepp extend function) and the lattice-based motion-planning algorithm ARA*. Fig. 9 shows that PRM* using deterministic Halton sampling slightly outperforms PRM* using random uniform sampling with respect to both the path length and curvature, while the lattice-based motion-planning algorithm ARA* outperforms both of them significantly.

D. Different Optimization Criteria

Bench-MR allows us to compare different optimization criteria. As an example, we compare PRM* with different cost functions, namely path length, minimum clearing distance and normalized curvature (also for this example we use the Reeds-Shepp extend function). Fig. 10 shows that maximizing the minimum clearing distance indeed increases the clearance compared to minimizing the path length or normalized curvature but also increases the number of cusps substantially. Minimizing the normalized curvature indeed decreases the curvature slightly compared to minimizing the path length. However, we found it difficult to minimize the normalized curvature in OMPL since its cost interface does not allow one to take the cusps into account that are created when connecting two edges. The right-most subfigure in Fig. 10 (top) shows that this limitation can create unexpected cusps.

E. Different Environment Complexities

Bench-MR allows us to compare motion-planning systems in procedurally-generated environments of different complexities. As an example, we show how the number of cusps of the

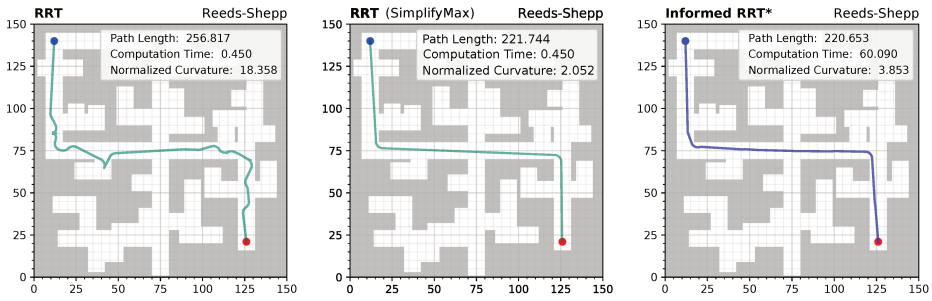


Fig. 7. Initial path of the the feasible motion-planning algorithm RRT using the Reeds-Shepp steer function after 0.45 seconds (left), its improvement using the post-smoothing algorithm SimplifyMax after less than 1 millisecond (center) and the path of the asymptotically (near) optimal motion-planning algorithm Informed RRT* using the Reeds-Sheep steer function after 60 seconds (right).

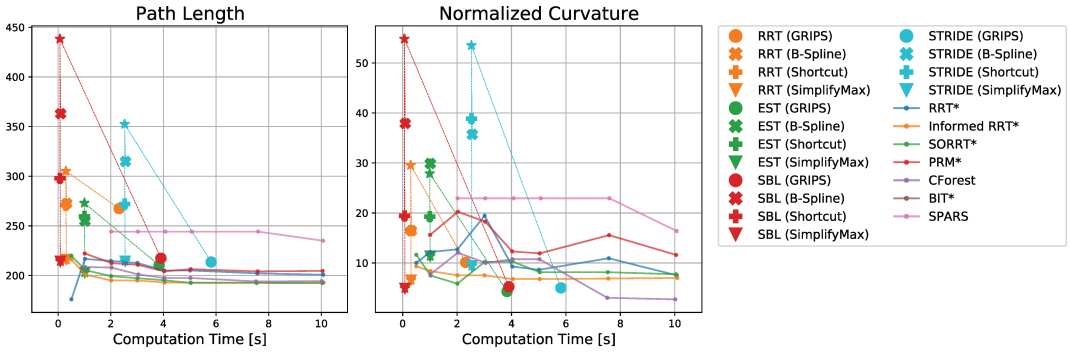


Fig. 8. Path length and normalized curvature of different combinations of the feasible motion-planning algorithms RRT, EST, SBL and STRIDE and post-smoothing algorithms compared against the asymptotically (near) optimal motion-planning algorithms RRT*, Informed RRT*, SORRT*, PRM*, CForest, BIT* and SPARS. Both performance metrics are reported with computation time limits of 0.5, 1.0, 2.0, 3.0, 4.0, 5.0, 7.5 and 10.0 seconds each in random indoor-like grid-based environments of size 150×150 cells and a desired minimum corridor width of 5 cells. The initial paths of the feasible motion-planning algorithms are marked with ★, and the paths of the post-smoothing algorithms are marked with ● for GRIPS, ✖ for B-Spline, ✚ for Shortcut and ▼ for SimplifyMax. The paths of the asymptotically (near) optimal motion-planning algorithms are solid lines marked with ·.

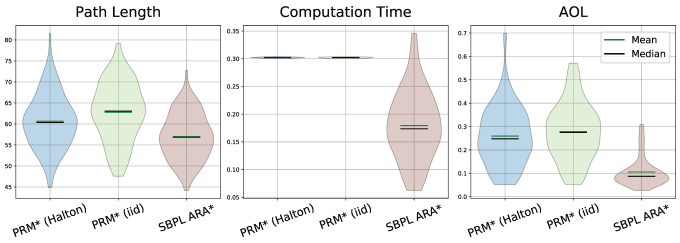


Fig. 9. Path length, computation time and AOL for different sampling strategies, namely PRM* using deterministic Halton sampling (left), PRM* using random uniform sampling (center) and the lattice-based motion-planning algorithm ARA* (right). All performance metrics are reported with a computation time limit of 0.3 seconds each (to be fair to the fast ARA*) in 100 random indoor-like grid-based environments with a desired minimum corridor width of 3 cells.

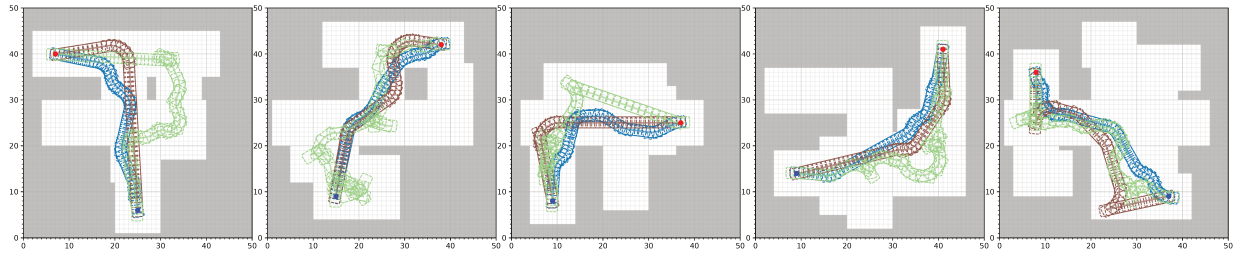
paths of different motion-planning algorithms using the Reeds-Shepp steer function varies with the desired minimum corridor width for random indoor-like grid-based environments and the desired percentage of blocked cells for random outdoor-like grid-based environments. Fig. 11 shows that the number of cusps significantly decreases for almost all motion-planning algorithms as the desired minimum corridor width increases. The number of cusps significantly increases for almost all motion-planning algorithms as the desired percentage of blocked cells increases.

F. Different Components of the Computation Time

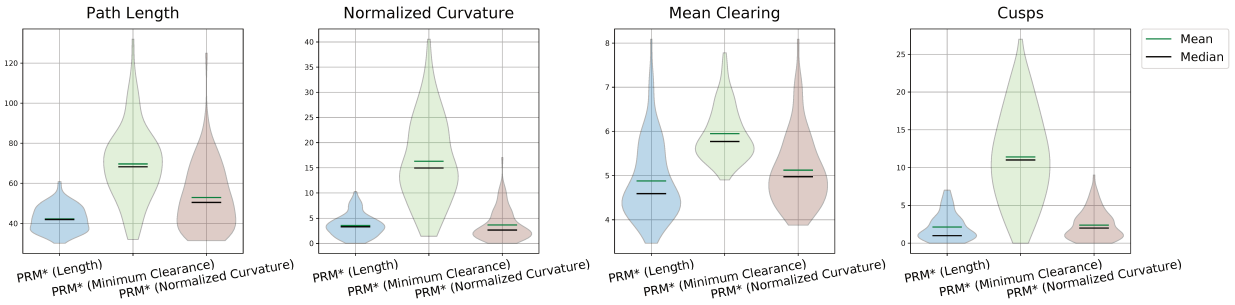
Bench-MR allows us to determine different components of the computation time. As an example, we determine the computation time needed for collision checking, Reeds-Shepp extend function evaluation and the remaining phases of motion planning. Fig. 12 shows the results for CForest, Informed RRT*, RRT and MHA*.

VII. Conclusions

Following the need for more reproducible evaluations of commonly used AI algorithms, and with the goal of comparing a large set of state-of-the-art motion planning techniques, we presented Bench-MR, the first open-source motion-planning benchmarking framework designed for sampling-based motion planning for nonholonomic, wheeled mobile robots. Unlike related software suites, Bench-MR is an easy-to-use and comprehensive benchmarking framework that aids practitioners and researchers in designing, testing and evaluating motion-planning systems and comparing them against the state of the art on complex navigation scenarios with many performance metrics. We presented several experiments that showed how Bench-MR can be used to understand the behavior of different motion-planning systems. The large variation in experimental results demonstrated that the performance of motion-planning systems depends on their components and that benchmarking



(a) Paths.



(b) Path length, normalized curvature, mean clearing distance and number of cusps.

Fig. 10. Results for different optimization criteria, namely PRM* with minimizing path length (left), maximizing minimum clearing distance (center) and minimizing normalized curvature (right). The colors of the paths (top) correspond to the colors of the optimization criteria (bottom). All metrics have been computed with a time limit of 2 seconds each in 100 random indoor-like grid-based environments with a desired minimum corridor width of 5 cells.

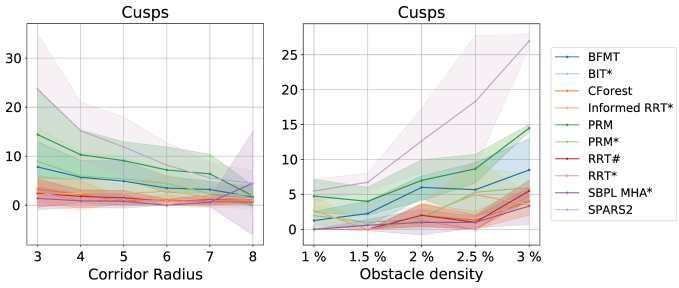


Fig. 11. Number of cusps for BFMT, BIT*, CForest, Informated RTT*, PRM, PRM*, RRT#, RRT*, MHA* and SPARS2 using the Reeds-Shepp steer function with a computation time limit of 15 seconds each in 5 random indoor-like grid-based environments of size 100×100 cells and desired minimum corridor widths ranging from 3 to 8 cells in increments of 1 cell (left) and 5 random outdoor-like grid-based environments of size 100 cells and desired percentages of blocked cells ranging from 1.0 to 3.0 percent in increments of 0.5 percent (right).

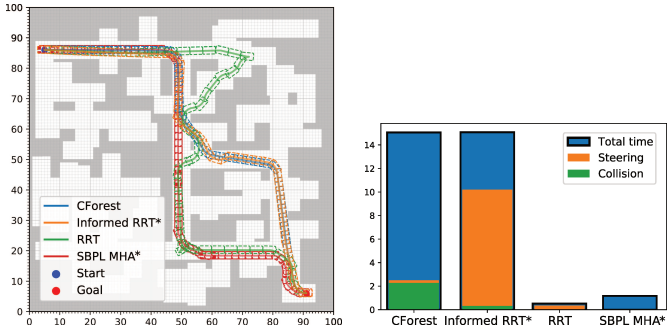


Fig. 12. Total computation time and its components for collision checking and extend function evaluation (that is, steering) for CForest, Informed RRT*, RRT and MHA*.

frameworks like Bench-MR are therefore vital for designing them for given applications and for guiding further research on motion planning. In future work, we plan to extend Bench-MR to dynamic environments.

ACKNOWLEDGMENTS

The authors thank Ziang Liu for his contributions to the software repository and testing of various algorithms. This work was supported by a Google Ph.D. Fellowship, the European Union's Horizon 2020 research and innovation program under grant agreement No. 732737 (ILIAD), and the US National Science Foundation (NSF) under grant numbers 1409987, 1724392, 1817189, 1837779 and 1935712.

REFERENCES

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[2] N. R. Sturtevant, "Benchmarks for grid-based pathfinding," *Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144 – 148, 2012, https://movingai.com/benchmarks/grids.html.

[3] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of SLAM algorithms," *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, 2009, http://ais.informatik.uni-freiburg.de/slamevaluation/datasets.php.

[4] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.

[5] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," *Advances in Neural Information Processing Systems*, vol. 16, pp. 767–774, 2003.

[6] L. Palmieri and K. O. Arras, "A novel RRT extend function for efficient and smooth mobile robot motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 205–211.

[7] T. Fraichard and A. Scheuer, "From Reeds and Shepp's to continuous-curvature paths," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 1025–1035, 2004.

[8] J. Luo and K. Hauser, "An empirical study of optimal motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1761–1768.

[9] M. Moll, I. A. Şucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 96–102, 2015.

[10] B. Cohen, I. A. Şucan, and S. Chitta, "A generic infrastructure for benchmarking motion planners," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 589–595.

[11] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 719–726.

[12] "Algorithms and Applications Group motion planning benchmark," https://parasollab.web.illinois.edu/resources/mpbenchmarks/.

[13] "PathBench: a benchmarking platform for classic and learned path planning algorithms," https://github.com/perfectly-balanced/PathBench.

[14] D. Calisi and D. Nardi, "Performance evaluation of pure-motion tasks for mobile robots with respect to world models," *Autonomous Robots*, vol. 27, no. 4, pp. 465–481, 2009.

[15] J. Weisz, Y. Huang, F. Lier, S. Sethumadhavan, and P. Allen, "RoboBench: Towards sustainable robotics system benchmarking," in *2016 IEEE International Conference on Robotics and Automation*, 2016, pp. 3383–3389.

[16] I. Rañó and J. Minguez, "Steps toward the automatic evaluation of robot obstacle avoidance algorithms," in *Workshop of Benchmarking in Robotics in IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 90–91.

[17] C. Sprunk, J. Röwekämper, G. Parent, L. Spinello, G. D. Tipaldi, W. Burgard, and M. Jalobeanu, "An experimental protocol for benchmarking robotic indoor navigation," in *Experimental Robotics*, 2016, pp. 487–504.

[18] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime Dynamic A*: An anytime, replanning algorithm." in *International Conference on Automated Planning and Scheduling*, 2005, pp. 262–271.

[19] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[20] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[21] L. Janson, B. Ichter, and M. Pavone, "Deterministic sampling-based motion planning: Optimality, complexity, and performance," *International Journal of Robotics Research*, vol. 37, no. 1, pp. 46–61, 2018.

[22] L. Palmieri, L. Bruns, M. Meurer, and K. O. Arras, "Dispertio: Optimal sampling for safe deterministic motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 362–368, 2019.

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

[24] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[25] A. Dobson, A. Krontiris, and K. E. Bekris, "Sparse roadmap spanners," in *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 279–296.

[26] T. Kunz and M. Stilman, "Kinodynamic RRTs with fixed time step and best-input extension are not probabilistically complete," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 233–244.

[27] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of International Conference on Robotics and Automation*. IEEE, 1997.

[28] G. Sánchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Robotics research*. Springer, 2003.

[29] B. Gipson, M. Moll, and L. E. Kavraki, "Resolution independent density estimation for motion planning in high-dimensional spaces," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013.

[30] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[31] J. A. Starek, J. V. Gomez, E. Schmerling, L. Janson, L. Moreno, and M. Pavone, "An asymptotically-optimal sampling-based algorithm for bi-directional motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 2072–2078.

[32] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013.

[33] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2997–3004.

[34] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Informed sampling for asymptotically optimal path planning," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 966–984, 2018.

[35] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 3067–3074.

[36] M. Otte and N. Correll, "C-FOREST: Parallel shortest path planning with superlinear speedup," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 798–806, 2013.

[37] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[38] A. Dobson and K. E. Bekris, "Improving sparse roadmap spanners," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013.

[39] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[40] F. Islam, V. Narayanan, and M. Likhachev, "Dynamic multi-heuristic A*," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 2376–2382.

[41] J. Van Den Berg, R. Shah, A. Huang, and K. Goldberg, "Anytime nonparametric A*," in *AAAI Conference on Artificial Intelligence*, 2011, p. 105–111.

[42] J.-P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," in *Robot Motion Planning and Control*. Springer, 1998, pp. 1–53.

[43] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[44] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[45] H. Banzhaf, L. Palmieri, D. Nienhüser, T. Schamm, S. Knoop, and J. M. Zöllner, "Hybrid curvature steer: A novel extend function for sampling-based nonholonomic motion planning in tight environments," in *International Conference on Intelligent Transportation Systems*, 2017, pp. 1–8.

[46] L. Palmieri, S. Koenig, and K. O. Arras, "RRT-based nonholonomic motion planning using any-angle path biasing," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 2775–2781.

[47] S. Gottschalk, "Separating axis theorem," Department of Computer Science, UNC Chapel Hill, Tech. Rep. TR96-024, 1996.

[48] E. Heiden, L. Palmieri, S. Koenig, K. O. Arras, and G. S. Sukhatme, "Gradient-informed path smoothing for wheeled mobile robots," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 1710–1717.