Benchmarking Sampling-based Motion Planning Pipelines for Wheeled Mobile Robots

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

[1] Department of Computer Science, University of Southern California, Los Angeles, USA
[2] Robert Bosch GmbH, Corporate Research, Stuttgart, Germany
[3] Division of Robotics, Perception and Learning (RPL), KTH Royal Institute of Technology, Stockholm, Sweden
heiden@usc.edu, Luigi.Palmieri@de.bosch.com, leonardb@kth.se

Abstract

Sampling-based motion planning is a key tool for several autonomous systems ranging from autonomous driving to service and intralogistic robotics. Over the past decades, several algorithms, extend functions and post-smoothing techniques have been introduced for such systems. Choosing the best combination of such components for an autonomous system's application is a tedious exercise, even for expert users. With the aim of helping researchers and practitioners in efficiently solving this issue, we have recently 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 provides a large variety of sampling-based motion-planning algorithms, extend functions, collision checkers, post-smoothing algorithms and optimization criteria. In this workshop paper, we complement our previous publication, by providing several examples on how to use it, together with the details on the framework architecture and components.

Introduction

In this paper we present **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. This work has previously been published at the International Conference on Robotics and Automation (Heiden et al. 2021)[1].

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.

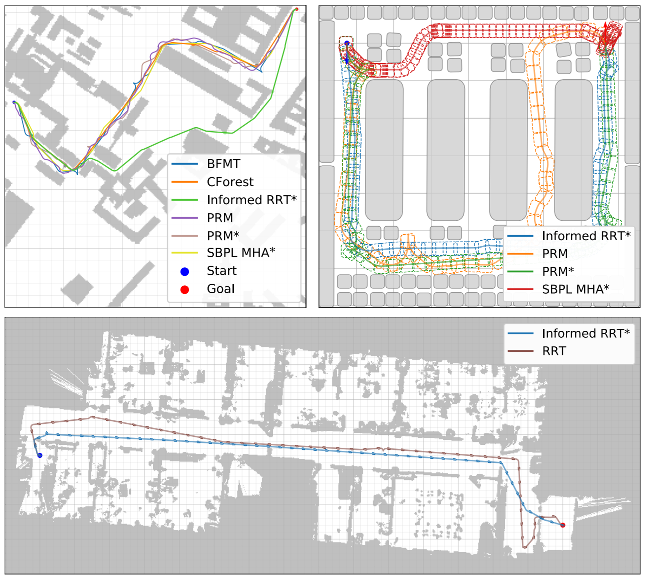

Figure 1: Selection of environments provided by Bench-MR: City grid from the Moving AI path-finding benchmark (Sturtevant 2012) (top left), polygon-based warehouse environment (top right), and thresholded occupancy grid from the Freiburg SLAM dataset (Kümmerle et al. 2009) (bottom).

[*]Equal contribution

[†]G.S. Sukhatme holds concurrent appointments as a Professor at USC and as an Amazon Scholar. This paper describes work performed at USC and is not associated with Amazon.

[1]We publish code and documentation on our Bench-MR website at https://robot-motion.github.io/bench-mr.

Much of Bench-MR builds on the Open Motion Planning Library (OMPL) (Şucan, Moll, and Kavraki 2012), but we also provide interfaces to other implementations of motion-planning algorithms (such as SBPL planners (Likhachev, Gordon, and Thrun 2003)) and extend functions (such as POSQ (Palmieri and Arras 2014) and continuous-curvature steering (Fraichard and Scheuer 2004)) 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.

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 (Sturtevant 2012) 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. (Luo and Hauser 2014) 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. (Moll, Şucan, and Kavraki 2015) have introduced a general benchmarking framework for motion-planning algorithms that is highly coupled with OMPL. It is highly customizable but lacks 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. (Cohen, Şucan, and Chitta 2012), different classes of motion-planning algorithms, including lattice-based planners.

Althoff et al. (Althoff, Koschi, and Manzinger 2017) 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 (Amato, Rauchwerger, and Morales 2013) provides several benchmarks for different robotic systems but contains only a small number of navigation scenarios for wheeled mobile robots. Instead Path-Bench (Clair et al. 2021) 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 (Calisi and Nardi 2009; Weisz et al. 2016; Rañó and Minguez 2006; Sprunk et al. 2016) 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.

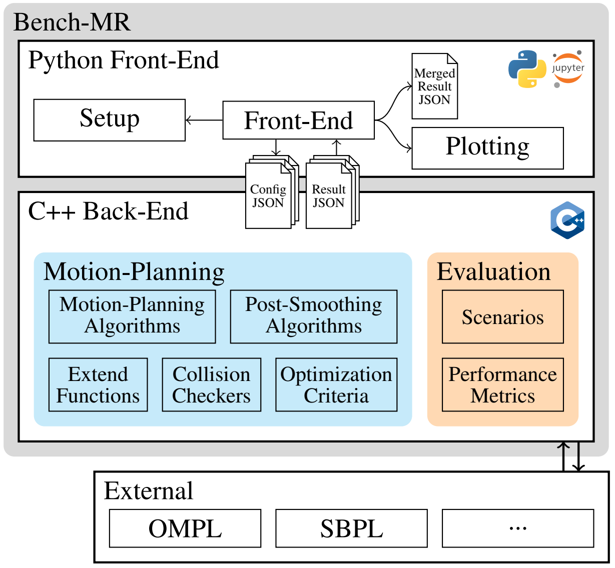

Figure 2: Architecture of Bench-MR. The components necessary for motion planning are shown in the box on the left (turquoise), and the utilities used in the evaluation are shown in the box on the right (orange). The implementation is split into a C++ back-end for running the performance-critical motion-planning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

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 (Şucan, Moll, and Kavraki 2012; Likhachev et al. 2005; Fraichard and Scheuer 2004). 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.

Bench-MR provides interfaces to two existing open-source motion-planning libraries, namely OMPL (Şucan, Moll, and Kavraki 2012) and SBPL (Likhachev, Gordon, and Thrun 2003), 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.

Bench-MR Planning Components

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

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 (LaValle, Branicky, and Lindemann 2004; LaValle 2006; Janson, Ichter, and Pavone 2018)): *feasible planners*, *asymptotically (near) optimal planners* and *lattice-based planners*.[2] 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 (Janson, Ichter, and Pavone 2018; LaValle, Branicky, and Lindemann 2004; Palmieri et al. 2019). We choose the most prominent open-source implementation for each class.

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 (LaValle and Kuffner Jr 2001), PRM (Kavraki et al. 1996), SPARS (Dobson, Krontiris, and Bekris 2013), RRT (LaValle and Kuffner Jr 2001; Kunz and Stilman 2015) using random forward propagation, EST (Hsu, Latombe, and Motwani 1997), SBL (Sánchez and Latombe 2003) and STRIDE (Gipson, Moll, and Kavraki 2013)).

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* (Karaman and Frazzoli 2011), BFMT (Starek et al. 2015), RRT# (Arslan and Tsiotras 2013)), informed search-based planners (such as Informed RRT* (Gammell, Srinivasa, and Barfoot 2014), SORRT* (Gammell, Barfoot, and Srinivasa 2018) and BIT* (Gammell, Srinivasa, and Barfoot 2015)), CForest (Otte and Correll 2013) and near-optimal planners (such as SST (Li, Littlefield, and Bekris 2016), an asymptotically near-optimal incremental version of RRT, SPARS (Dobson, Krontiris, and Bekris 2013) and SPARS2 (Dobson and Bekris 2013)).

Lattice-Based Planners Lattice-based planners use state lattices with predefined motion primitives that encode differential constraints (Pivtoraiko, Knepper, and Kelly 2009). Bench-MR currently provides lattice-based planners from SBPL (such as ARA* (Likhachev, Gordon, and Thrun 2003), AD* (Likhachev et al. 2005), MHA* (Islam, Narayanan, and Likhachev 2015) and ANA* (Van Den Berg et al. 2011)).

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

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 (Laumond, Sekhavat, and Lamiraux 1998) to connect two given robot configurations exactly for a given steer function. We refer the reader to (Kunz and Stilman 2015) 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.

Robot Dynamics Models Our software includes 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 (Paden et al. 2016).

Steer Functions Several common steer functions, namely Dubins (Dubins 1957), Reeds-Shepp (Reeds and Shepp 1990), Continuous Curvature (Banzhaf et al. 2017; Fraichard and Scheuer 2004) and POSQ (Palmieri and Arras 2014; Palmieri, Koenig, and Arras 2016) are included.

Motion Primitives Bench-MR provides a few motion primitives from SBPL, and further models can be added via the motion primitive file interface of SBPL.

Collision Checkers

Bench-MR includes 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. Furthermore, we provide a two-dimensional polygon-based approach to collision checking, which uses the *separating axis theorem* (Gottschalk 1996) 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.

Post-Smoothing Algorithms

Bench-MR includes several post-smoothing algorithms from OMPL, such as B-Spline, Shortcut and Simplify-Max (Şucan, Moll, and Kavraki 2012). It also includes the recently introduced GRradient-Informed Post Smoothing (GRIPS) algorithm (Heiden et al. 2018), a hybrid approach that combines short-cutting with locally optimized waypoint placement based on the distance field of the environment.

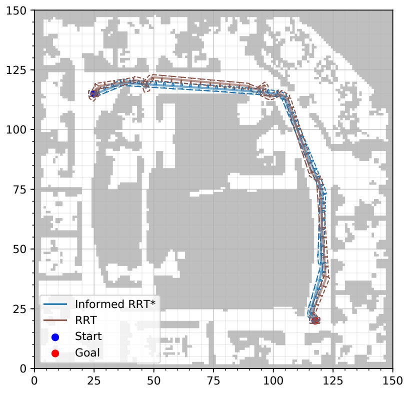

Figure 3: Predefined grid-based environment obtained from a gray-scale image of an Intel office building (Kümmerle et al. 2009).

Optimization Criteria

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

Bench-MR Evaluation Components

In the following, we explain the Bench-MR evaluation components.

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.

Predefined Grid-Based Environments We provide two classes of predefined grid-based environments. First, we include a selection of city grids from the Moving AI pathfinding benchmark (Sturtevant 2012), 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). Second, Bench-MR also provides image-based grids that can be created from grey-scale images by thresholding with a user-defined occupancy cutoff value (a common representation for maps generated by SLAM algorithms (Kümmerle et al. 2009)). Examples are shown in Fig. 1 (bottom) and Fig. 3.

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 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).

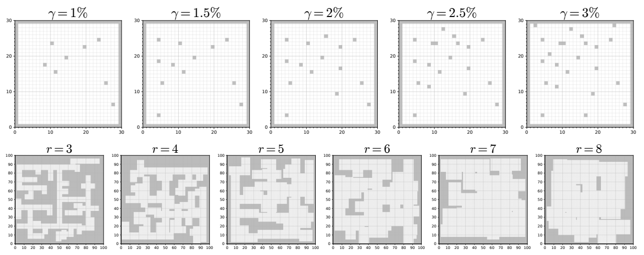

Figure 4: Procedurally-generated grid-based environments, namely random outdoor-like environments with different percentages of blocked cells (top), and random indoor-like environments with different minimum corridor widths (bottom).

Predefined Polygon-Based Environments Bench-MR includes 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.

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.

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.

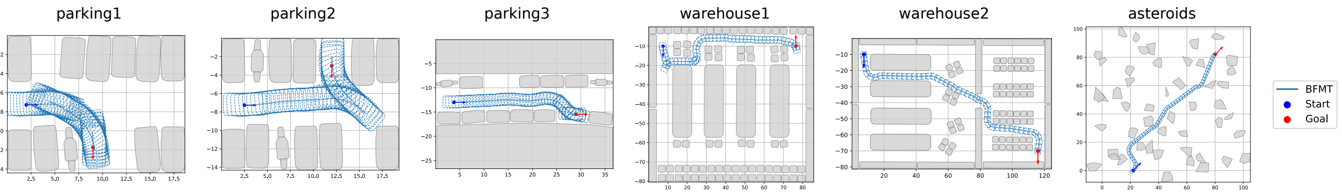

Figure 5: Paths for polygon-based environments computed by the *Bidirectional Asymptotically Optimal Fast Marching Tree* (BFMT) motion-planning algorithm using the Reeds-Shepp steer function. The first five environments are predefined, and the right-most environment is procedurally generated.

3. *The maximum curvature* (κ_{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).
6. *The number of cusps* (Banzhaf et al. 2017) measures how often a robot has to stop on a path and turns its wheels to abruptly change its heading.

Example Usage

In the following, we demonstrate how Bench-MR allows for an easy benchmarking of different planners. For more experiments that give insights on the interplay between various components of the planning pipeline, we refer the reader to our main paper (Heiden et al. 2021).

Introductory Example

As initial step we set our motion planning benchmark object mpb, associated to a configuration file in the JSON format that contains all the benchmark configurations. We specify a generator for procedural grid environments that resemble indoor-like spaces with corridors that have a width of 3 cells:

```python
from mpb import MPB
mpb = MPB(config_file = "benchmark_template.json")
mpb.set_corridor_grid_env(radius = 3)
```

Next, we define the planning algorithms that we wish to compare, the type of steer function, and the number of runs to execute each combination of planning algorithm and steer function:

```python
mpb.set_planners(["rrt", "rrt_star", "informed_rrt_star"])
mpb.set_steer_functions(["reeds_shepp"])
mpb.run(runs=3)
mpb.visualize_trajectories()
```

To visualize the resulting paths (see Fig. 6), we need to call a single function on our motion planning benchmark instance mpb:

```python
mpb.visualize_trajectories()
```

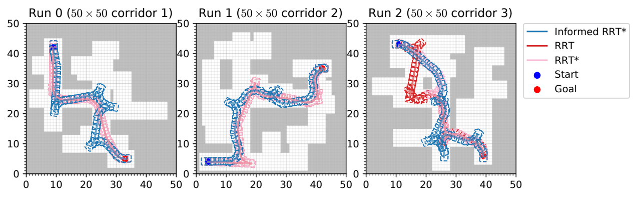

Figure 6: Paths obtained from the listed example.

To visualize the recorded metrics and gather statistical insights (see Fig. 7), we can call the following function:

```python
mpb.plot_planner_stats()
```

As shown in Fig. 7, it visualizes the performance on key metrics as violin plots across the 3 runs that have been executed, grouped by the motion planning algorithms.

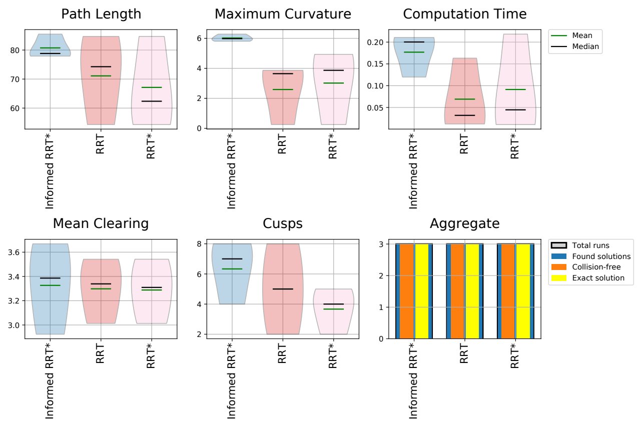

Figure 7: Statistics obtained with the illustrated example.

Parallel Benchmark Execution

Bench-MR supports multi-processing out of the box to distribute benchmarks across multiple processors. After the parallel execution, the results from the separately evaluated benchmark instances can be merged for further analysis.

In the following example we vary the time allotment within which the planner Informed RRT* equipped with the Reeds-Shepp steer function has to find a solution. Therefore, we create three `MPB` instances with different planning times (0.5 s, 1 s and 10 s), and select the corridor-like procedural grid generator as envionment. Finally, the `MPB` instances are added to a `MultipleMPB` object which executes the three benchmarks in parallel over five runs each.

```
from mpb import MultipleMPB, MPB
pool = MultipleMPB()
for time in [0.5, 1, 10]:
    m = MPB()
    m["max_planning_time"] = time
    m.set_corridor_grid_env()
    m.set_planners(["informed_rrt_star"])
    m.set_steer_functions(["reeds_shepp"])
     pool.benchmarks.append(m)
pool.run_parallel("test_parallel", runs=5)
```

Comparison: Random and Halton Sequences, State Lattice

In this example we show how to compare different sampling strategies, namely an i.i.d. uniform random distribution, Halton sequence, and state lattice.

We start by configuring a benchmark using a corridor environment generated with a radius of 3 cells, choose the PRM* planning algorithm and the extend function Reeds-Shepp:

```
from mpb import MPB
import matplotlib as mpl
mpb = MPB()
mpb.set_planners(["prm_star"])
mpb.set_steer_functions(["reeds_shepp"])
mpb.set_corridor_grid_env(radius=3.0)
mpb["ompl.seed"] = 1
mpb["ompl.cost_threshold"] = 0
mpb["max_planning_time"] = 0.3
```

Next, we create three copies of the benchmark object `mpb` and assign to them the different sampling strategies. For the state lattice we need to also choose a different planning algorithm, as it has to be a method from SBPL (we choose ARA* here):

```
from copy import deepcopy
mpb_iid = deepcopy(mpb)
mpb_iid.set_id("iid")
mpb_iid["ompl.sampler"] = "iid"

mpb_halton = deepcopy(mpb)
mpb_halton.set_id("halton")
mpb_halton["ompl.sampler"] = "halton"

mpb_statelattice = deepcopy(mpb)
mpb_statelattice.set_id("sbpl_arastar")
mpb_statelattice.set_planners(["sbpl_arastar"])
mpb_statelattice["sbpl.scaling"] = 1
mpb_statelattice["sbpl.resolution"] = 0.25
mpb_statelattice["max_planning_time"] = 1
```

Finally, we run the previously defined benchmarks in parallel:

```
pool = MultipleMPB()
pool.benchmarks.append(mpb_iid)
pool.benchmarks.append(mpb_halton)
pool.benchmarks.append(mpb_statelattice)
pool.run_parallel(runs=100, id="exp", show_plot=False)
pool.merge("exp/exp.json",
    plan_names=["PRM* (iid)", "PRM* (Halton)", "SL (ARA*)"])
```

The results have been merged to the previously configured results JSON file `exp/exp.json` which we can use to plot statistical results from the previous runs, see Fig. 8:

```
from plot_stats import plot_planner_stats
plot_planner_stats("exp/exp.json",
    metrics="path_length, planning_time, aol",
        save_file="./sampling.pdf",
            num_colors=4, ticks_rotation=20)
```

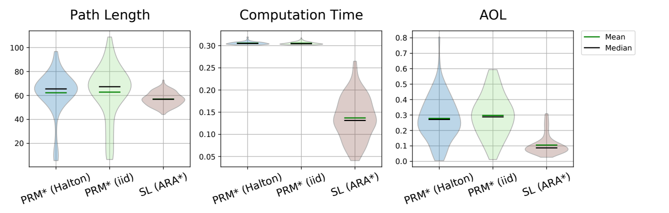

Figure 8: Statistics obtained by comparing different sampling strategies.

Metrics

In this example we show the available metrics that our benchmarking suite offers, how we can evaluate and visualize them.

We start by setting our planning context. Again, we adopt the corridor environment and use the steer function Reeds-Shepp to compare three algorithms from OMPL: RRT, RRT* and Informed RRT*.

```
mpb = MPB()
mpb.set_corridor_grid_env(radius = 3)
mpb.set_planners(["rrt", "rrt_star", "informed_rrt_star"])
mpb.set_steer_functions(["reeds_shepp"])
# optional run ID, number of runs (environments)
mpb.run(id="test_run", runs=3)
```

We can plot all the available statistics with the following command:

```
mpb.plot_planner_stats(", ".join(stat_names.keys()))
```

Note that all the available metrics are stored in the `stat_names` dictionary which maps from the name of the metric to its printable title which is used for plotting purposes.

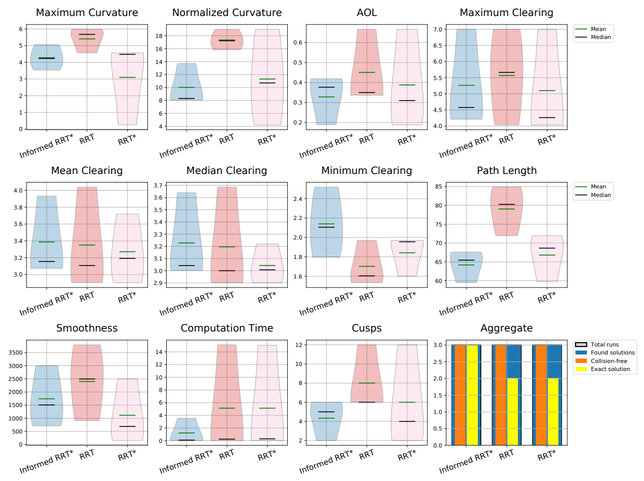

Figure 9: Metrics obtained by comparing different planners.

Additionally we can plot also the time spent in different planning phases (e.g. steer function, collision checking), see Fig. 10:

```
mpb.plot_planner_timings()
```

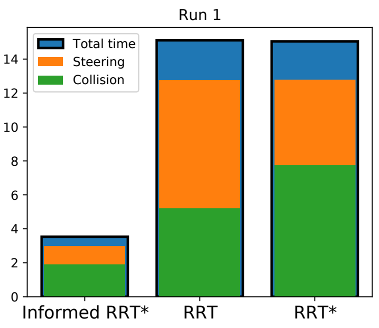

Figure 10: Example computation times reported for a single run, separated by the time used for computing the extend function (steering) and for collision checking.

Conclusions

Bench-MR is an easy-to-use and comprehensive benchmarking framework that aids practitioners and researchers in designing, testing and evaluating motion-planning systems. Various motion planning components can be easily compared against the state of the art on complex navigation scenarios with many performance metrics. Unlike other benchmarking tools, our suite of motion planning components is particularly tailored to applications in wheeled mobile robotics, and provides a productive user interface. In this workshop paper that complements our previous work (Heiden et al. 2021), apart from reporting details on the framework and its components, we have presented several examples that demonstrated how to use Bench-MR. In future work, we plan to extend Bench-MR to dynamic environments to support more realistic scenarios in autonomous driving.

Acknowledgments

We 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. 101017274 (DARKO), and the US National Science Foundation (NSF) under grant numbers 1409987, 1724392, 1817189, 1837779 and 1935712.

References

Althoff, M.; Koschi, M.; and Manzinger, S. 2017. CommonRoad: Composable benchmarks for motion planning on roads. In *IEEE Intelligent Vehicles Symposium (IV)*, 719–726.

Amato, N.; Rauchwerger, L.; and Morales, M. 2013. Algorithms and Applications Group motion planning benchmark. https://parasollab.web.illinois.edu/resources/mpbenchmarks/.

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

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

Calisi, D.; and Nardi, D. 2009. Performance evaluation of pure-motion tasks for mobile robots with respect to world models. *Autonomous Robots* 27(4): 465–481.

Clair, J.; Milenkova, R.; Shields, A.; Yao, J.; Patel, Z.; Hu, D.; Kelly, P.; and Saeedi, S. 2021. PathBench3D: A Benchmarking Platform for Classic and Learned Path Planning Algorithms. https://github.com/perfectly-balanced/PathBench.

Cohen, B.; Şucan, I. A.; and Chitta, S. 2012. A generic infrastructure for benchmarking motion planners. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 589–595.

Dobson, A.; and Bekris, K. E. 2013. Improving sparse roadmap spanners. In *2013 IEEE International Conference on Robotics and Automation*. IEEE.

Dobson, A.; Krontiris, A.; and Bekris, K. E. 2013. Sparse roadmap spanners. In *Algorithmic Foundations of Robotics X*, 279–296. Springer.

Dubins, L. E. 1957. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics* 79(3): 497–516.

Fraichard, T.; and Scheuer, A. 2004. From Reeds and Shepp's to continuous-curvature paths. *IEEE Transactions on Robotics* 20(6): 1025–1035.

Gammell, J. D.; Barfoot, T. D.; and Srinivasa, S. S. 2018. Informed sampling for asymptotically optimal path planning. *IEEE Transactions on Robotics* 34(4): 966–984.

Gammell, J. D.; Srinivasa, S. S.; and Barfoot, T. D. 2014. 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*, 2997–3004.

Gammell, J. D.; Srinivasa, S. S.; and Barfoot, T. D. 2015. 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*, 3067–3074.

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

Gottschalk, S. 1996. Separating axis theorem. Technical Report TR96-024, Department of Computer Science, UNC Chapel Hill.

Heiden, E.; Palmieri, L.; Bruns, L.; Arras, K. O.; Sukhatme, G. S.; and Koenig, S. 2021. Bench-MR: A Motion Planning Benchmark for Wheeled Mobile Robots. *IEEE Robotics and Automation Letters* 6(3): 4536–4543.

Heiden, E.; Palmieri, L.; Koenig, S.; Arras, K. O.; and Sukhatme, G. S. 2018. Gradient-Informed Path Smoothing for Wheeled Mobile Robots. In *IEEE International Conference on Robotics and Automation*, 1710–1717.

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

Islam, F.; Narayanan, V.; and Likhachev, M. 2015. Dynamic multi-heuristic A*. In *IEEE International Conference on Robotics and Automation*, 2376–2382.

Janson, L.; Ichter, B.; and Pavone, M. 2018. Deterministic sampling-based motion planning: Optimality, complexity, and performance. *International Journal of Robotics Research* 37(1): 46–61.

Karaman, S.; and Frazzoli, E. 2011. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research* 30(7): 846–894.

Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; and Overmars, M. H. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4): 566–580.

Kümmerle, R.; Steder, B.; Dornhege, C.; Ruhnke, M.; Grisetti, G.; Stachniss, C.; and Kleiner, A. 2009. On measuring the accuracy of SLAM algorithms. *Autonomous Robots* 27(4): 387–407. http://ais.informatik.uni-freiburg.de/slamevaluation/datasets.php.

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

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

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

LaValle, S. M.; Branicky, M. S.; and Lindemann, S. R. 2004. On the Relationship between Classical Grid Search and Probabilistic Roadmaps. *International Journal of Robotics Research* 23(7-8): 673–692.

LaValle, S. M.; and Kuffner Jr, J. J. 2001. Randomized kinodynamic planning. *International Journal of Robotics Research* 20(5): 378–400.

Li, Y.; Littlefield, Z.; and Bekris, K. E. 2016. Asymptotically optimal sampling-based kinodynamic planning. *International Journal of Robotics Research* 35(5): 528–564.

Likhachev, M.; Ferguson, D. I.; Gordon, G. J.; Stentz, A.; and Thrun, S. 2005. Anytime Dynamic A*: An Anytime, Replanning Algorithm. In *International Conference on Automated Planning and Scheduling*, 262–271.

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

Luo, J.; and Hauser, K. 2014. An empirical study of optimal motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1761–1768.

Moll, M.; Şucan, I. A.; and Kavraki, L. E. 2015. Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization. *IEEE Robotics & Automation Magazine* 22(3): 96–102.

Otte, M.; and Correll, N. 2013. C-FOREST: Parallel shortest path planning with superlinear speedup. *IEEE Transactions on Robotics* 29(3): 798–806.

Paden, B.; Čáp, M.; Yong, S. Z.; Yershov, D.; and Frazzoli, E. 2016. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles* 1(1): 33–55.

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

Palmieri, L.; Bruns, L.; Meurer, M.; and Arras, K. O. 2019. Dispertio: Optimal sampling for safe deterministic motion planning. *IEEE Robotics and Automation Letters* 5(2): 362–368.

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

Pivtoraiko, M.; Knepper, R. A.; and Kelly, A. 2009. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics* 26(3): 308–333.

Rañó, I.; and Minguez, J. 2006. 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*, 90–91.

Reeds, J.; and Shepp, L. 1990. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics* 145(2): 367–393.

Sánchez, G.; and Latombe, J.-C. 2003. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *Robotics research*. Springer.

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

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

Sturtevant, N. R. 2012. Benchmarks for grid-Based pathfinding. *Transactions on Computational Intelligence and AI in Games* 4(2): 144 – 148. https://movingai.com/benchmarks/grids.html.

Şucan, I. A.; Moll, M.; and Kavraki, L. E. 2012. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine* 19(4): 72–82. doi:10.1109/MRA.2012.2205651. https://ompl.kavrakilab.org.

Van Den Berg, J.; Shah, R.; Huang, A.; and Goldberg, K. 2011. Anytime nonparametric A*. In *AAAI Conference on Artificial Intelligence*, 105–111.

Weisz, J.; Huang, Y.; Lier, F.; Sethumadhavan, S.; and Allen, P. 2016. RoboBench: Towards sustainable robotics system benchmarking. In *2016 IEEE International Conference on Robotics and Automation*, 3383–3389.