Journal of Intelligent & Robotic Systems
https://doi.org/10.1007/s10846-019-01086-y

Quantitative and Qualitative Evaluation of ROS-Enabled Local and Global Planners in 2D Static Environments

Alexandros Filotheou[1] · Emmanouil Tsardoulias[1] · Antonis Dimitriou[1] · Andreas Symeonidis[1] · Loukas Petrou[1]

Received: 21 March 2019 / Accepted: 22 August 2019

Abstract

Apart from perception, one of the most fundamental aspects of an autonomous mobile robot is the ability to adequately and safely traverse the environment it operates in. This ability is called Navigation and is performed in a two- or three-dimensional fashion, except for cases where the robot is neither a ground vehicle nor articulated (e.g. robotics arms). The planning part of navigation comprises a global planner, suitable for generating a path from an initial to a target pose, and a local planner tasked with traversing the aforementioned path while dealing with environmental, sensorial and motion uncertainties. However, the task of selecting the optimal global and/or local planner combination is quite hard since no research provides insight on which is best regarding the domain and planner limitations. In this context, current work performs a comparative analysis on qualitative and quantitative aspects of the most common ROS-enabled global and local planners for robots operating in two-dimensional static environments, on the basis of mission-centered and planner-related metrics, optimality and traversability aspects, as well as non-measurable aspects, such as documentation quality, parameterisability, ease of use, etc.

Keywords Path planning · Path traversing · Local planners · Global planners · Unmanned Ground Vehicles · ROS · RFID localization

1 Introduction

Nowadays we are experiencing a significant penetration of Radio Frequency Identification (RFID) technology in the market, mainly in supply chain management (retail, warehouses), health-care, and banking, establishing an 11.1 billion market in 2017 [1]. In the same time, "robotics" experience their own share of success in the so-called "4th Industrial Revolution", where a fusion of interconnected technologies can communicate, have access to big data, and take decisions, thereby improving the efficiency of systems. Merging of these two technologies enables robots to interact with RFID-tagged objects. Such merging includes the domains of robot navigation [2, 3] and automated inventorying [4–6]. In this context, we are building RFID-enabled, autonomous, unsupervised robots, capable of performing continuous inventorying and accurate localisation of RFID tags. The robots are equipped with depth cameras, lidar sensors, UHF RFID readers, and UHF RFID antennas, and are capable of performing Simultaneous Localisation (of their own poses) and Mapping (of the environment), as well as identifying and locating RFID-tagged products inside the environment by exploiting state-of-the-art localisation algorithms [7, 8]. The robots could be deployed inside warehouses or large retail stores, possibly in the presence of people. The sub-problems for the successful completion of our goal are (*i*) the employment of the appropriate path-planning and path-traversing strategy, in order for the robot to be able to successfully navigate within its surrounding space,

✉ Alexandros Filotheou
alexandros.filotheou@issel.ee.auth.gr

Emmanouil Tsardoulias
etsardou@eng.auth.gr

Antonis Dimitriou
antodimi@auth.gr

Andreas Symeonidis
asymeon@eng.auth.gr

Loukas Petrou
loukas@eng.auth.gr

[1] School of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece

Published online: 21 October 2019

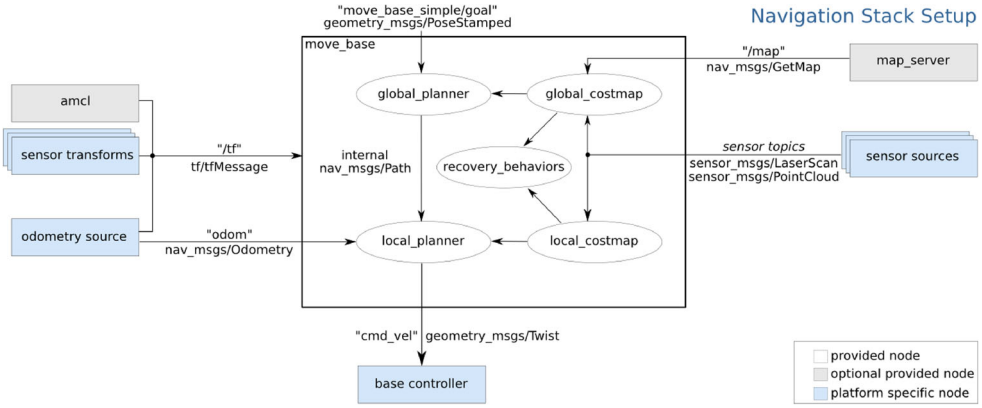

Fig. 1 Overview of the move_base package. Image taken from `move_base`'s website

considering the generated map of the environment and the obstacles in its operating area, (*ii*) the generation of the accurate and undistorted map of the environment and the accurate localisation of the robot within it and (*iii*) the localisation of the RFID-tagged objects in the previously created map. This work is focused in the first subproblem, i.e. the selection of the appropriate path-planning and path-traversing strategy for 2D environmental representations.

For an autonomous ground robot to traverse its environment, several steps have to be performed. First, a target-selector algorithm should be employed, uptaking the task of calculating/generating a target for the robot to reach. This target is usually the desired pose of the robot, either in the $2D$ ($[x, y, \theta]$), or in the $N-$dimensional space if the robot is, for instance, articulated. Next, an algorithm capable of taking as input the robotic perception of the world (usually a $2D$ or $3D$ map), as well as the current and goal robot poses must be employed. Then, a geometric path is generated, which, if followed, will lead the robot from its current pose to the goal pose. This path is generated by a **global (path-) planner**, which deals with the global information of the problem (i.e. the entirety of the map). Finally, a **local planner** is needed, which takes the global path and the *local perception* of the robot as inputs, and generates motor velocities, so as to guide the robot into following the global path, while at the same time doing so as safely as possible (assuming that the robot should avoid collisions with stationary or moving objects, although this is not always the case [9]). By local perception we denote the live raw sensory data representing the dynamic alterations of the environment, in contrast to the static overall map.

Currently, ROS (Robot Operating System)[1], released in 2009 [10], is one of the dominant robotic middleware platforms. ROS is described as a meta-operating system and is designed to offer hardware abstraction along with standardised module-based development over a structured communications layer. One of the main strengths of ROS is the standardisation of message types, using a simple language-neutral IDL (Interface Definition Language) to describe them, resulting in a theoretically language-agnostic software implementation. This standardisation allows for the development of decoupled ROS packages, i.e. easily reusable collections of *nodes* that offer specific functionality. A node is a computer process that performs computations. Nodes are combined together into a graph and communicate with one another using streaming topics and/or Remote Procedure Call services.[2] In ROS, a robotic system usually comprises several nodes. For example, in Fig. 1, `map_server` provides the map in which the robot is located, `odometry source` provides odometric information, the `amcl` node performs localisation, outputting to `move_base` the relation between the map's and the robot's odometric frames of reference, a `sensor transform` node provides the relation between two frames of reference (for example the relation between the robot's range finder frame of reference and the robot's base link), `sensor sources` provide `move_base` with readings from sensors, and `base controller` takes as input the motor velocity commands computed by `move_base` and applies them to the robot's motors. ROS' hardware requirements are minimal in processing power and memory, making it able of being executed in a Raspberry Pi or a BeagleBone, nevertheless its actual requirements vary depending on the number and type of nodes a robot is tasked to run.

ROS has become widely popular in the robotics community, since it offers an abundance of free-to-use packages by research teams across the globe. It already offers navigation capabilities in the form of *stacks* (collections of ROS packages), the most well-known and commonly-used of them being **`move_base`**, which internally employs a global and a local planner. Furthermore, `move_base` is famous for its plug-and-

[1] http://www.ros.org/

[2] http://wiki.ros.org/Nodes

play capabilities, but at the same time notorious for the abundance of parameters it contains, as well as for the not-so-obvious local navigation decisions it makes (e.g. the decisions and actions it internally performs when it perceives that the robot is stuck).

The goal of this work is to delve into the depths of the `move_base` package and evaluate different ROS-enabled global and local planners so as to investigate which combination performs better when tested under the simple task of 2D traversability in heterogeneous environments represented by occupancy-grid maps. The paper is structured as follows: Section 2 presents the state-of-the art in robotic planning, describing theoretic approaches on global and local planners, whereas Section 3 focuses on the practical side of ROS-enabled planning packages, including an overview of `move_base`'s functionality, as well as the most common ROS global and local planners. Next, Section 4 presents the experimental formulation and methodology of evaluation, along with the metrics used for evaluating each category (global and local planners, plus their combinations), followed by Section 5 which discusses the results. Finally, Section 6 presents the article's conclusions and Appendix holds a detailed account and analysis of the results of the conducted simulations and real-world experiments.

2 State of the Art on Robotic Planners

Even though typical *robot navigation* includes a global planner for generating a path from a start pose to a goal pose and a local planner for traversing this path, the usual nomenclature may vary, since a global planner can be referred to as *Path-planner* and a local planner as *Navigator*.

2.1 Global Planners

As far as global path planners are concerned, a plethora of approaches suitable for $2D$ environment representations to $N-$dimensional spaces have been proposed throughout the years. Usually, global path-planning techniques belong in one out of six main algorithmic families, namely *Visibility Graphs*, *Skeletonisation-based techniques*, *Probabilistic Roadmaps*, *Rapidly exploring Random Trees*, *State Lattices*, and *Navigation Functions*.

Visibility Graphs were one of the first path-planning methods proposed by Losano-Perez and Wesley in 1979 [11], describing a path-creation algorithm in a convex environment, where the obstacles are transformed so as to depict untraversable areas due to the geometric constraints of the robot's footprint. Then, the visibility graph containing the transformed obstacles' vertices as nodes is created. Finally, a search algorithm is applied to generate the final path. As described in [12] and [13], visibility graphs suffer from high requirements in computational resources and obstacle geometric restrictions in order to correctly operate, thus other approaches have been proposed to deal with these issues [14].

Among skeletonisation techniques, the Generalised Voronoi Diagram (GVD) is the dominant algorithm employed in order to produce a skeleton of the free space the robot operates in. Examples of GVD employment in global path-planning techniques are mentioned in [15], where GVD is used to generate a collision-free and smooth path, in [16], where GVD is followed by the application of the Fast Marching algorithm to minimise the length of the path, and in [17], which introduces the Voronoi Uncertainty Fields (VUFs), combining GVD as a global planner and a local planner to navigate the vehicle.

One of the most famous path-planning algorithmic families are Probabilistic Roadmaps (PRMs). Their concept is simple: sampling is performed in the environment's free/unoccupied space and a graph whose edges are safe for traversing is created. Then, a search algorithm is applied in the graph to find the minimum cost path. PRMs were initially introduced by Kavraki et al. in [18], nevertheless several alterations have been proposed, such as [19] where the visibility graph concepts are used to enhance the PRM graph, [20] which introduces the Lazy PRM algorithm that dynamically minimises the graph's connections, and [21] where the Hybrid PRM is proposed, combining different PRMs according to the environmental properties.

Another global path-planning methodology is that of the Rapidly exploring Random Trees algorithms (RRTs), initially proposed by La Valle in 1998 [22]. RRTs iteratively generate tree-like structures, initiating from a root-node and terminating when a leaf reaches the desired goal. Several alterations exist, such as the Execution Extended RRT (ERRT) [23], bidirectional RRTs [24], RRT* [25], Cell-RRT [26], and T-RRT [27], among others.

Path-planning in state lattices emerged in 2005 by Pivtoraiko and Kelly [28]. A state lattice is a search space, comprising a discretised set of a system's (the robot's kinematic model) reachable configurations which can encode feasibly-traversable paths. The paths are formed by local connections between constraint-compliant states. After the development of the search space, a set of spatially distinct path primitives are generated; this space encodes the local connections and eliminates redundancies so that a planning query on the connected search graph can be efficiently executed.

Navigation functions are a special class of potential functions [29] for the navigation of mobile robots. Potential functions assume a known map; they assign a potential value on every point (in landmark-based maps) or grid cell (in grid-based maps), with each of them having a higher

potential value the lower its distance from an obstacle. Conversely, the goal position/configuration is assigned a low potential value. The potential field principle is attractive due to its simplicity and elegance, however a number of substantial shortcomings were reported through the years [30, 31], such as the susceptibility of making a robot getting trapped at local minima and the rise of oscillations when a robot gets near obstacles or narrow passages. Navigation functions try to overcome these issues, being functions (a) for which the goal's potential is assigned zero value, or, if the goal is unreachable, infinite value, and (b) that have a monotonic gradient, i.e no local minima exists except at the goal. However, the navigation function method may exhibit slow convergence, especially when the robot's environment includes narrow passages, thus it requires tailored tuning [32]. Furthermore, in high-dimensional spaces, where the robot's or obstacles' shapes are complex, the computational cost rises sharply [33].

2.2 Local Planners

Once the path has been generated using a global planner, a local planner must be deployed in order for the robot to follow the global plan and make sure it avoids both static and dynamic obstacles.

One of the oldest local planners is VFH (Vector Field Histograms), proposed in 1991 by Borenstein and Koren [34]. VFHs generate a polar histogram, assigning each angle with probability of that direction being occupied, relative to the robot's orientation. Then, sufficiently large openings for the robot to navigate safely through are identified and a cost function for each opening is calculated, ultimately selecting the one with the smallest cost. Improvements on the VFH are the VFH+, incorporating arc-like local trajectories in contrast to VFH's straight lines [35] and VFH* which verifies that a candidate direction guides the robot around the obstacle, using the A* algorithm and appropriate cost and heuristic functions [36].

Another famous approach is DWA (Dynamic Window Approach), suggested by Fox, Burgard and Thrun [37]. DWA samples the local environment with possible trajectories directly derived from the motion dynamics of the robot, calculating a cost for each sample. Then, the velocities set that maximises an objective function is selected for application. This function includes the robot's heading with respect to the target, the clearance of the closest obstacle on the trajectory and the previous linear-velocity in order to account for the body's inertia.

GNTs (Gap Navigation Trees) are tree-like structures generated from online robot sensor measurements, encoding paths from the current robot pose to any place in the environment [38]. A GNT is updated as the robot moves and produces optimal paths if the environment is simply-connected, provided that the environmental boundaries are smooth, since the GNT tries to identify "gaps" in the sensor measurements.

Another strand of local planners initiated in 2004 with the proposition of the Nearness Diagram navigation approach (ND) by Minguez and Montano [39]. The ND methodology initially generates two nearness diagrams: the PND (from the central robot Point) and the RND (from the Robot) to represent information about the obstacles' proximity. Both PND and RND are further analysed and special safety sections and gaps are calculated, based on which the robot is assigned a safety situation status among five different ones. Ultimately, five motion laws are evaluated according to the safety class of the robot, resulting in the proper velocity command of the robot at that specific time. In [40] the ND+ methodology is proposed, adding a sixth scenario to balance the division of motion laws, increasing the smoothness of transitions between some of the scenarios. Finally, SND (Smooth Nearness-Diagram) navigation is an evolution of ND+, where a single motion law, applicable to all possible configurations of surrounding obstacles is proposed, removing the abrupt behavior transitions when the robot navigates close to obstacles.

The Elastic Band approach by Quinlan and Khatib [41] bridges path planning and control theory: based on a global plan, the local planner produces a deformable path in real-time, so that changes in the environment (detected by sensors), uncertainties in measurement, model uncertainties or moving objects are incorporated into the robot's path-planning and path-following. To achieve its objectives (one of which is to preserve the global nature of the planned path), the approach builds on artificial forces: predefined internal forces contract the path and make it smoother, while external forces maintain separation from obstacles. However, the original approach does not explicitly incorporate temporal or kinodynamic constraints. An extension of the original approach, deforming trajectories rather than paths, is presented in [42].

The Timed-Elastic-Band approach [43] on the other hand, inspired by the idea of the Elastic Band method, *does* take into account both temporal and kinodynamic constrains. The original approach provides a real-time online trajectory planner for differential-drive robots. Mimicking a predictive controller, it reformulates the trajectory planning and the control inputs as an optimisation problem subject to kinodynamic and obstacle-avoidance constraints, while simultaneously taking temporal information into account. An extension of the Timed-Elastic-Band is presented in [44], where a more generic formulation is introduced, extending its model support to ackerman steering models, while making motion reversals possible (i.e. making possible for a robot with a car-like kinematics model to park autonomously).

3 ROS Enabled Global and Local Planners

As aforementioned, ROS offers a navigation stack which essentially takes information from odometry, sensor streams and a goal pose and outputs safe velocity commands to a mobile base.[3] The ROS package at the core of the navigation stack is `move_base`, a high-level architectural diagram of which is depicted in Fig. 1.

When seen as a black box, `move_base` outputs velocity commands to a robot's motors and assumes the existence of the following inputs, either in the form of ROS messages (structured data) or transformations (relations between frames of reference):

- the robot's estimated pose in the form of a transformation between the robot's odometric frame of reference and the map's frame of reference, provided here by the `amcl`[4] ROS package. AMCL stands for Adaptive Monte Carlo Localisation [45] and is currently the de facto localiser in the ROS ecosystem
- transformations between coordinate frames of the robot's sensors and effectors using ROS' own transformation mechanism (`tf`)
- odometry information and (optionally) a map of the environment
- distance data from either a range sensor, or a sensor that can output point clouds, such as a depth camera

Furthermore, ROS offers an environmental representation called a *costmap*, including information about the traversability of the world, based on static and dynamic obstacles (assuming it is more expensive to move close to obstacles), as well as the footprint of the robot. When seen as a white box, `move_base` includes:

- A *global costmap* that is static and has the same size as the map.
- A *local costmap* which is generated online in the vicinity of the robot and is based on the sensory input in order to deal with static and dynamic obstacles.
- A *global planner* taking as input a goal and the global costmap and producing a global, geometric path.
- A *local planner* that takes as input the global path and the local costmap and calculates velocity commands.
- A module called `recovery_behaviours` that takes both costmaps as input, identifies when the robot cannot progress with the desired velocities, and applies predefined sets of motions, aiming at "un-stucking" the robot. These actions are triggered each time (a) the robot is perceived to be oscillating, (b) a global plan has not been received for some amount of time, or (c) the local planner has failed to procure valid velocity commands for a set amount of time.

  Specifically, `move_base` applies two kinds of recovery behaviours: (a) a 360-degree rotation that aims at clearing the local costmap of any spurious measurements (falsely-positive perceived obstacles) and (b) a costmap reset that clears the navigation stack's costmaps by reverting them to the static map outside of a given radius away from the robot[5]. The latter is usually employed multiple times and in a hierarchical manner, starting from some radius within the local costmap's semiwidth and moving in closer towards the robot's footprint. If the robot is still perceived as stuck after executing all predefined recovery behaviours, navigation is aborted and the robot halts its motion, at least until a new goal is provided.

3.1 Global Planners

This section provides a brief overview of the global planners considered in this review.

3.1.1 navfn

The `navfn`[6] ROS package is based on the *NF1* navigation function approach [29]. It provides a fast interpolated function that can be utilised to produce plans for a mobile base, assumed to have a circular footprint. The navigation function takes as input the global costmap, a start and an end point, and produces the minimum-cost plan from start to end by employing the Dijkstra or A* search algorithms. The main disadvantages of NF1 (and therefore of `navfn`) are the lack of smoothness of the produced paths, since these are constituted of straight-line segments joined by integer multiples of $\pi/4$ angles, and, most importantly, that NF1 produces paths that graze obstacles [46]. As a ROS software package, it is compatible with the latest stable version of ROS (`melodic`) at the time of writing of this article.

3.1.2 global_planner

The `global_planner` package provides an implementation of a fast, interpolated global planner for navigation[7]. It was designed as a flexible successor of `navfn` and is able to generate paths using either the A* or Dijkstra's algorithms, so that the computational load can be lowered (the latter's is greater than the former's), although the produced paths are not considered optimal in the 8-connected sense. As a

[3] http://wiki.ros.org/navigation

[4] http://wiki.ros.org/amcl

[5] http://wiki.ros.org/clear_costmap_recovery

[6] http://wiki.ros.org/navfn

[7] http://wiki.ros.org/global_planner

ROS software package, it is compatible with the latest stable version of ROS.

3.1.3 asr_navfn

The `asr_navfn`[8] package essentially operates in exactly the same way as `navfn`, with the added benefit that, in case the desired goal is infeasible (meaning within or too close to an obstacle), it computes a feasible "nearest goal" to the commanded one. As a ROS software package, it is not compatible with the latest stable version of ROS.

3.1.4 MoveIt!

In contrast to `navfn` and the majority of the other global planners presented here, `MoveIt!` does not come as a `move_base` plugin [47]. Its main limitations with respect to path planning for mobile robots are that (a) it is primarily targeted at and developed for robotic manipulators, i.e. robots with (multiple-joint) arms, (b) it cannot plan for multi-degree-of-freedom joints, i.e. it can only be used for planning the motion of holonomic mobile robots, and (c) it requires the transformation of costmaps into its own OMPL state-space. As far as planners are concerned, `MoveIt!` can internally utilise OMPL (Open Motion Planning Library[9]), STOMP (Stochastic Trajectory Optimisation for Motion Planning[10]), SBPL (Search-Based Planning Library[11]) or CHOMP (Covariant Hamiltonian Optimisation for Motion Planning[12]). As a ROS software package, it is compatible with the latest stable version of ROS.

3.1.5 sbpl_lattice_planner

The `sbpl_lattice_planner`[13] global planner is a state lattice approach, utilising the `SBPL` library. In stark contrast to all other global planners mentioned in this article, paths are generated by combining a series of motion primitives, valid motions based on the robot's kinematic model. Among its advantages are that (a) it takes account of the robot's kinematic model so that the produced path is actually feasible by a local planner and (b) it provides the ability of weighing motions: depending on what motions are preferable (for instance when turning, the engineer can choose whether it is more desirable for the robot to turn in-place or plan ahead so that it traverses an arc), undesired motions are penalised so that the robot's trajectory is tuned to fit given specifications, if any. Planning is performed in the x, y, θ dimensions and takes the robot orientation into account. Finally, the low-level ARA* [48] or AD* [49] planners are used to create the global path. As a ROS software package, it is compatible with the latest stable version of ROS.

3.1.6 sbpl_dynamic_env_global_planner

The `sbpl_dynamic_env_global_planner`[14] [50] is similar to `sbpl_lattice_planner`, nevertheless it can incorporate information both from the static costmap and the predicted future trajectories of moving obstacles. This is done by grouping spatio-temporal information of where and when the motion will be safe and the planning is performed in the standard three spatial dimensions, including a fourth one concerning safety [51]. As a ROS software package, it is not compatible with the latest stable version of ROS.

3.1.7 lattice_planner

The `lattice_planner`[15] package provides a `move_base` global planner plugin for a time-bounded A* state lattice planner. This planner utilises the ROS costmap structure and can produce time dependent, dynamically feasible navigation paths for robots with differential drive constraints. As a ROS software package, it is not compatible with the latest stable version of ROS.

3.1.8 waypoint_global_planner

Another ROS global planner is the `waypoint_global_planner`.[16] This planner takes as input *manually*-inserted waypoints (therefore engineers must themselves take caution to provide feasible poses as the planner has no knowledge of obstacles in the map) and generates a path that traverses them in sequence. The path is comprised of line segments connecting one input point to the next and the final pose of the robot adopts the orientation of the line segment connecting the second-to-last and last path points. As a ROS software package, it is not compatible with the latest stable version of ROS.

3.1.9 voronoi_planner

The `voronoi_planner`[17] package creates a global plan from a point to another by using the GVD (Generalised

[8] http://wiki.ros.org/asr_navfn

[9] http://ompl.kavrakilab.org/

[10] http://wiki.ros.org/stomp_motion_planner

[11] http://wiki.ros.org/sbpl

[12] http://www.nathanratliff.com/thesis-research/chomp

[13] http://wiki.ros.org/sbpl_lattice_planner

[14] http://wiki.ros.org/sbpl_dynamic_env_global_planner

[15] https://github.com/marinaKollmitz/lattice_planner

[16] https://github.com/gkouros/waypoint-global-planner

[17] http://wiki.ros.org/voronoi_planner

Voronoi Diagram) of the environment. The GVD is constructed from the costmaps' obstacles and, as aforementioned, contains all points that are equidistant from the two closest obstacles, providing a skeleton of the free space. The final global path is restricted to exist in the GVD (as opposed to the planners that use search algorithms such as A*). Of course, this ability forces to planner to generate safer but non-optimal in length paths. As a ROS software package, it is compatible with the latest stable version of ROS.

It should be noted that among all global planners considered in this subsection, only `sbpl_dynamic_env_global_planner` is able to take account of moving objects in the robot's operating environment, that is, estimate their motion and project it to the future given their pose and velocity. The rest can only function by regarding them as stationary for the time between two consecutive global path generations.

An interesting comment on the ROS-available global planners is that almost all of them are somewhat naive when time and resources are concerned. For example, almost all global planners use alterations of the well-known A* search algorithm, which, even though it's length-optimal, it can be very slow when the map is large. Next, the available local planners are described.

3.2 Local Planners

In contrast to the global planners, only a few available and commonly used local planners exist.

3.2.1 dwa_local_planner

The `dwa_local_planner`[18] is based on the work of Fox et al. [37]. As explained earlier, DWA discretely samples in the robot's control space, performs forward simulation for each sample, evaluates each trajectory against the local costmap, discards the illegal trajectories and finally selects the highest-scoring trajectory that satisfies both the robot's kinematic restrictions and traversability safety. `dwa_local_planner` does not support obstacle avoidance for moving obstacles. As a ROS software package, it is compatible with the latest stable version of ROS (`melodic`) at the time of writing of this article.

3.2.2 eband_local_planner

The `eband_local_planner`[19] is based on the elastic bands theory. An elastic band is a deformable collision-free path generated by a global path, incorporating the information of obstacle proximity. This deformation from the global path is performed during runtime, as the local perception changes. `eband_local_planner` does not support obstacle avoidance for moving obstacles and is not compatible with the latest stable version of ROS.

3.2.3 teb_local_planner

The `teb_local_planner`[20] package implements an online local trajectory planner for control and navigation of mobile robots. What this local planner does is to get the path generated by a global planner and minimise it during runtime with respect to trajectory execution time, separation from obstacles and compliance with kinodynamic constraints, such as satisfying maximum velocities and accelerations. This planner supports non-holonomic robots as well (differential drive and Ackerman-steering cars) and is based on the theoretical work presented in [43], which improves the elastic band theory. `teb_local_planner` *does* support obstacle avoidance for moving obstacles. As a ROS software package, it is compatible with the latest stable version of ROS.

After briefly describing the ROS-available global and local planners, it is time to provide the experimental formulation, i.e. in what environments the algorithms were tested, under what conditions and, most importantly, which evaluation metrics are employed.

4 Benchmarking Setup & Metrics Used

4.1 Methodology of Evaluation, Environmental Setup, and Notation

The evaluation of all combinations of planners considered in this document is performed both in simulated and real environments. The robot used in all simulated and real conditions is the second version of Turtlebot,[21] a differential-drive robot with a circular footprint of radius $r = 0.2$m. The two simulated environments all planners were benchmarked are readily-available worlds of the Gazebo simulator. These worlds properly simulate most of the conditions a $2D$ mobile base would face in a static, indoor environment: corridors of different widths, open narrow passages where satisfaction of constraints is critical and easier to be violated while testing the ability of local planners to procure and execute fine control maneuvers, U-turns, multiple consecutive turns, and obstacles that the local planner must negotiate on the robot's way to the goal. The real environment where planners were tested is the

[18] http://wiki.ros.org/dwa_local_planner

[19] http://wiki.ros.org/eband_local_planner

[20] http://wiki.ros.org/teb_local_planner

[21] https://www.turtlebot.com/turtlebot2/

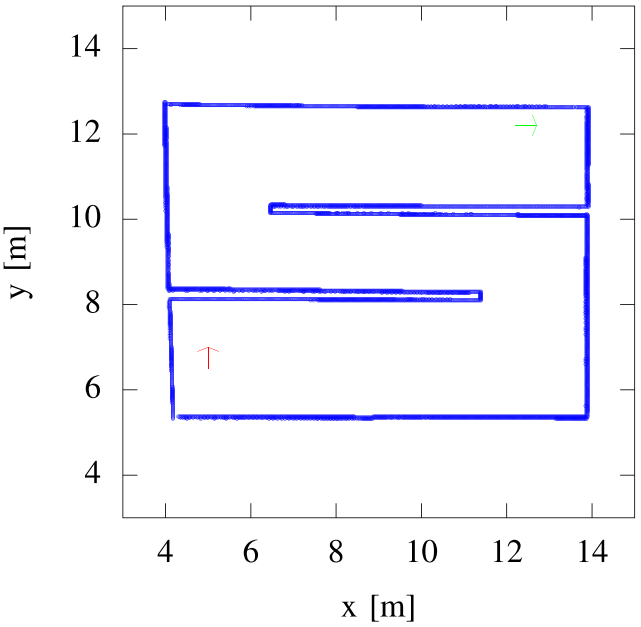

Fig. 2 Map of the CORRIDOR gazebo world. The green arrow (top right) signifies the robot's initial pose p_0^C; the red one (bottom left) the goal pose p_G^C

Computer Systems Architecture Laboratory (CSAL) of the Electrical Engineering department of AUTh.

Figure 2 depicts the map of simulated world CORRIDOR, denoted hereafter by $\boldsymbol{M}_C$, Fig. 3 depicts a section from a significantly larger in size map called WILLOWGARAGE, denoted by $\boldsymbol{M}_W$, and Fig. 4 depicts the map of CSAL, denoted by $\boldsymbol{M}_L$.

Green arrows signify the robot's initial pose, while red ones signify the target pose. The maps of the two simulated environments were constructed using ROS' SLAM package `gmapping`,[22] while that of the real environment was constructed using `open-karto`[23] and they were provided to the robot at the start of each simulation. While $\boldsymbol{M}_C$ is a map resembling the structure of a typical warehouse, map $\boldsymbol{M}_W$ is a floor plan resembling that of a typical office floor. The former's difficulty is significantly lower compared to that of the latter: corridors are wide, no narrow passages exist, and there are only two turns whose distance is sufficiently large so that it's expected that planners will easily steer the robot away from both wall-ends − although results show that even this expectation is optimistic for some combinations of planners.

The goals' poses were set as such in order to maximise the difficulty the planners would face in both finding a feasible path and traversing it and, therefore, they facilitate the exposition of the planners' shortcomings. For map CORRIDOR the robot's initial pose was $p_0^C \equiv$ (12.2m, 12.2m, 0.0 rad) and the goal pose was $p_G^C \equiv$ (5.0m, 6.5m, $\pi/2$ rad). For map WILLOWGARAGE $p_0^W \equiv$ (69.0m, 79.0m, 0.0 rad) and $p_G^W \equiv$ (58.0m, 45.0m, $\pi/2$ rad). Finally, for map CSAL $p_0^L \equiv$ (18.6m, 11.3m, 0.0 rad) and $p_G^L \equiv$ (11.3m, 2.86m, 0.0 rad).

Each global/local planner combination was tested in each environment with the same initial and goal poses for $N = 10$ times, and therefore the evaluation of the performance of all combinations was made using statistical

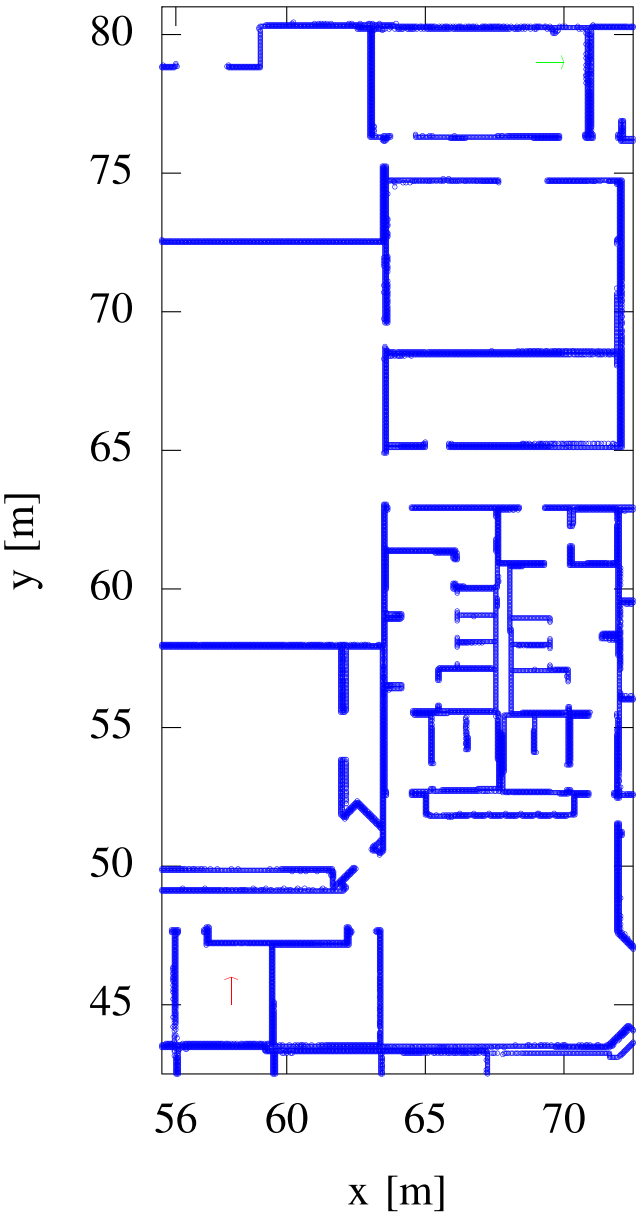

Fig. 3 Section of the map of the WILLOWGARAGE gazebo world. The green arrow (top right) signifies the robot's initial pose p_0^W; the red one (bottom left) the goal pose p_G^W

[22] https://openslam-org.github.io/gmapping.html

[23] http://wiki.ros.org/open_karto

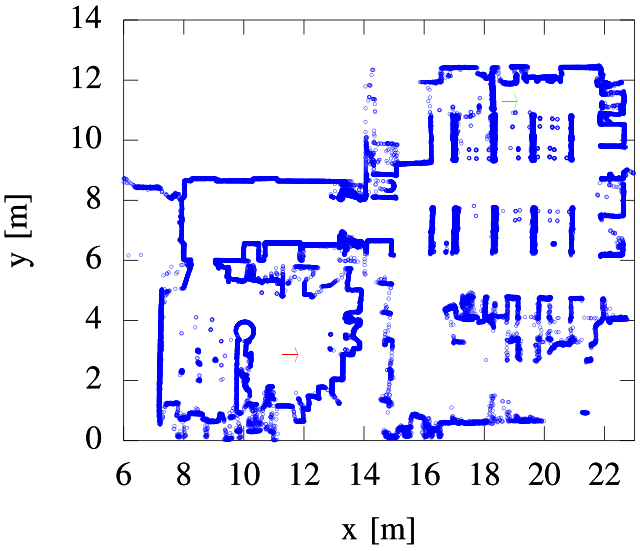

Fig. 4 Section of the map of the CSAL environment. The green arrow (top right) signifies the robot's initial pose p_0^L; the red one (bottom left) the goal pose p_G^L

means. Each combination was given a time-period to execute its navigation from start to goal, set at $t_C^{max} = 120$ sec for world CORRIDOR, $t_W^{max} = 180$ sec for world WILLOWGARAGE, and $t_L^{max} = 600$ sec in environment CSAL.

All simulations were performed in Linux Ubuntu 16.04, on a PC with a i7 CPU of 12 threads, 32GB of memory, and a clock frequency of 4.00 GHz. All experiments performed in environment CSAL were performed in Linux Ubuntu 16.04, on a PC with a i5 CPU of 4 threads, 8GB of memory, and a clock frequency of 3.20 GHz.

4.2 Path-related Definition of Metrics

In what follows we make the following assumptions and definitions: a **path** $\boldsymbol{P} : [1, n] \rightarrow \mathbb{R}^2 \times (-\pi, \pi]$ is a sequence of poses $p_i, i \in [1, 2, \ldots, n]$, i.e. $\boldsymbol{P} \equiv (p_1, p_2, \ldots, p_n)$, where $p_i = (x_i, y_i, \theta_i)$, i.e. the $x-$wise and $y-$wise coordinates of a point in $\mathbb{R}^2$ and θ_i is the orientation of a vector that originates at (x_i, y_i) with respect to the x axis. The cardinality of $\boldsymbol{P}$ may also be referred to as $|\boldsymbol{P}|$ and is equal to the number of poses in $\boldsymbol{P}$. A **collection of N paths** $\boldsymbol{P}_j$, $j \in [1, N]$ shall be denoted by $\mathcal{P} = \{\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots \boldsymbol{P}_N\}$. The **distance between two poses** p_i and p_j is selected to be the Euclidean distance $d(p_i, p_j) = ((x_i - x_j)^2 + (y_i - y_j)^2)^{1/2}$. An **occupancy grid map** $\boldsymbol{M} : [1, q] \rightarrow \mathbb{R}^2$ is a non-ordered set of points in the 2D Cartesian plane: $\boldsymbol{M} \equiv \{(x_1^M, y_1^M), (x_2^M, y_2^M), \ldots, (x_q^M, y_q^M)\}$. Furthermore, $\mu(x), \sigma(x)$ denote the mean value and standard deviation of variable x.

The **length** of a path $\boldsymbol{P}$, denoted by $l(\boldsymbol{P})$, is calculated as

$$l(\boldsymbol{P}) = \sum_{i=1}^{|\boldsymbol{P}|-1} d(p_i, p_{i+1}) \tag{1}$$

i.e. it is the sum of distances between consecutive poses p_i and p_{i+1}.

We denote the **smoothness of a path** $\boldsymbol{P}$ by $s(\boldsymbol{P})$, and define it as

$$s(\boldsymbol{P}) = \left(\frac{1}{|\boldsymbol{P}| - 2} \sum_{i=1}^{|\boldsymbol{P}|-1} (\theta_{i+1} - \theta_i)^2 \right)^{1/2} \tag{2}$$

The **mean minimum distance of a path** $\boldsymbol{P}$ **from obstacles in map** $\boldsymbol{M}$ is the average distance of the poses consisting the path to their closest obstacle. It is denoted by $d(\boldsymbol{P}, \boldsymbol{M})$, and defined as

$$d(\boldsymbol{P}, \boldsymbol{M}) = \frac{1}{|\boldsymbol{P}|} \sum_{k=1}^{|\boldsymbol{P}|} \min_{i=1,2,\ldots,q} d(p_k, m_i) \tag{3}$$

where $p_k \in \boldsymbol{P}$, $k = 1, 2, \ldots, |\boldsymbol{P}|$, and $m_i \in \boldsymbol{M}$, $i = 1, 2, \ldots, q$.

The **overall minimum distance of a collection of paths** $\mathcal{P}$ **to obstacles in** $\boldsymbol{M}$, denoted by $\inf(d(\mathcal{P}, \boldsymbol{M}))$ (so as to point to the absolute nature of this minimum value), is defined as

$$\inf(d(\mathcal{P}, \boldsymbol{M})) = \min_{j=1,2,\ldots,N} \left\{ \min_{\substack{k=1,2,\ldots,|\boldsymbol{P}_j| \\ i=1,2,\ldots,q}} d(p_k^j, m_i) \right\} \tag{4}$$

where $\boldsymbol{P}_j \in \mathcal{P}$, $p_k^j \in \boldsymbol{P}_j, k = 1, 2, \ldots, |\boldsymbol{P}_j|$, and $m_i \in \boldsymbol{M}$, $i = 1, 2, \ldots, q$.

The **mean deviation of a path** $\boldsymbol{P}_1$ **from a path** $\boldsymbol{P}_2$ is calculated as the mean distance of each pose consisting $\boldsymbol{P}_1$ to its nearest pose in $\boldsymbol{P}_2$:

$$d_\delta(\boldsymbol{P}_1, \boldsymbol{P}_2) = \frac{1}{|\boldsymbol{P}_1|} \sum_{k=1}^{|\boldsymbol{P}_1|} \min_{l=1,2,\ldots,|\boldsymbol{P}_2|} d(p_k, p_l) \tag{5}$$

where $p_k \in \boldsymbol{P}_1$, $k = 1, 2, \ldots, |\boldsymbol{P}_1|$, and $p_l \in \boldsymbol{P}_2$, $l = 1, 2, \ldots, |\boldsymbol{P}_2|$.

The **total deviation of a path** $\boldsymbol{P}_1$ **from a path** $\boldsymbol{P}_2$ is calculated as the sum of the distance of each pose consisting $\boldsymbol{P}_1$ to its nearest pose in $\boldsymbol{P}_2$:

$$d_\Delta(\boldsymbol{P}_1, \boldsymbol{P}_2) = |\boldsymbol{P}_1| \cdot d_\delta(\boldsymbol{P}_1, \boldsymbol{P}_2) \tag{6}$$

The Frechet distance metric, introduced by Maurice Frechet for continuous curves in a metric space in 1906 [52], is a measure of similarity between two curves. In this article's context, it is preferred to the Pompeiu-Hausdorff distance [53] due to the latter not accounting for the location and ordering of the points along a curve/path. For discrete curves $\boldsymbol{P}_1 : [1, m] \rightarrow V$, $\boldsymbol{P}_2 : [1, n] \rightarrow V$ (such as the actual path pose samples and global

plans) consisting of a sequence of endpoints $\lambda(\boldsymbol{P}_1) \equiv (\boldsymbol{P}_1(1), \boldsymbol{P}_1(2), \ldots \boldsymbol{P}_1(m)) \equiv (v_1, v_2, \ldots v_p)$ and $\lambda(\boldsymbol{P}_2) \equiv (\boldsymbol{P}_2(1), \boldsymbol{P}_2(2), \ldots \boldsymbol{P}_2(n)) \equiv (u_1, u_2, \ldots u_g)$ respectively, the **discrete Frechet distance** is defined by:

$$\delta_{dF}(\boldsymbol{P}_1, \boldsymbol{P}_2) = \min\{\|\boldsymbol{L}\| \mid \boldsymbol{L} \text{ is a coupling between } \boldsymbol{P}_1 \text{ and } \boldsymbol{P}_2\} \quad (7)$$

where $\|\boldsymbol{L}\| = \max_{i=1,\ldots q} d(u_{a_i}, u_{b_i})$. A coupling $\boldsymbol{L}$ is a sequence of distinct pairs from $\lambda(\boldsymbol{P}_1) \times \lambda(\boldsymbol{P}_2)$: $L \equiv ((u_{a_1}, v_{b_1}), (u_{a_2}, v_{b_2}), \ldots, (u_{a_q}, v_{b_q})$ and $d(a, b)$ is a metric of distance (here equal to the Euclidean distance as defined above) between points a and b.

Denoting by $\boldsymbol{\mathcal{G}}$ a collection of N global plans produced across N conducted simulations, $\boldsymbol{\mathcal{G}} = \{\boldsymbol{G}_1, \boldsymbol{G}_2, \ldots, \boldsymbol{G}_N\}$, and by $\boldsymbol{\mathcal{P}}$ a collection of N traversed paths $\boldsymbol{\mathcal{P}} = \{\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_N\}$, the evaluation metrics for global planners, local planners, and their combinations is featured in Table 1.

4.3 Overall Evaluation and Ranking

The overall evaluation of each combination of global and local planners shall be performed on the basis of the performance of its two components and on their joint performance. Each metric m described above shall be taken into account and assigned a weight $w_m \in \mathbb{R}_{\geq 0}$, so that generic evaluation is made feasible and tractable for variable specifications (depending on the application and the uncertainty of the robot's model, an engineer might prefer to assign further importance on, for instance, the length of the robot's traversed path than its overall minimum distance from obstacles). The utmost goal is the assignation of a scalar value to each combination of global and local planners that distinguishes them and ranks them based on their performance.

We initially make the assumption that the value of a combination of global and local planners should be strictly increasing, so that larger values reflect better performance. This value depends not only on the value of each metric hitherto discussed, but, more specifically, on the nature of the contribution of a metric. For instance travel times between $p_0^{\boldsymbol{M}_C}$ and $p_0^{\boldsymbol{M}_C}$ contribute more the lower their value is, but, with respect to the overall distance between obstacles, the value of a combination of planners should increase the greater that distance is. Therefore the value of a combination of planners depends on the proportionality contribution of each specific metric. Table 1 summarises the contribution of each metric that pertains to global planners, local planners, and their combinations. Arrows pointing upward indicate proportionality; arrows pointing downward indicate inverse proportionality. More details can be found in Appendix A.

Since what we seek is the assignation of a scalar value $V(C)$ to each combination of global and local planners C, we must pass through the construction of a valid value function V. Function V must (a) be strictly increasing (so that it accurately expresses the value of a combination based on performance metrics, and at the same time provide a meaning to their difference which can be traced back to the difference between their performance) and (b) account for metrics of different units of measurement. To this end, we begin by normalising the values of metrics within their respective interval of minimum and maximum values across combinations of global and local planners − so that the value of all metrics is expressed in the range of $[0, 1]$ (without a unit of measurement) − and depending on their context. The latter means that the value of, for instance, the mean length of N global plans will be expressed between the global mean minimum length and global mean maximum length of all combinations' global plans − since this metric is independent of the success or failure of the mission of a combination −, but the mean travel time between p_0^M and p_G^M for a map $\boldsymbol{M}$, which does depend on it, can only and therefore will only be expressed between the mean minimum and mean maximum travel times of combinations that succeeded in getting the robot from p_0^M and p_G^M. The normalising function for a metric m shall then be $N(m)$:

$$N(m) = \frac{m - \min m}{\max m - \min m} \quad (8)$$

Let S denote the set of combinations of global and local planners C that succeeded in making the robot travel from p_0^M to p_G^M for map $\boldsymbol{M}$. Let D denote the set of metrics that do not depend on the success of a mission (in the above sense), i.e. metrics concerning global and local planners but not their combination. Let also $I_A(x)$ denote the indicator function for a metric x and set A: $I_A(x)$ equals one if $x \in A$ and zero otherwise. Then the indicator function for combination C regarding metric m, $I(C, m) = I_S(C) \,||\, I_D(m)$ is zero when C was unsuccessful in its mission and m is a metric concerning the combination of global and local planners; in all other cases, $I(C, m)$ equals one. The formulation of this indicator function in such a way makes taking account of all metrics described feasible and the calculation of V tractable.

For global planners, local planners, or their combinations, if their value regarding metric m is proportional to the value of m (such as the value of the metric of the robot's overall minimum distance from obstacles), the value of a planner, or a combination of planners C, for metric m shall be expressed in the interval $\mathbb{R}_{\geq 0} \times [0, 1]$ and expressed by $V_q(C, m)$:

$$V_q(C, m) = I(C, m) \cdot w_m \cdot N(m) \quad (9)$$

Table 1 Evaluation metrics on global planners, local planners, and their combination (left), their description (middle), and the nature of their contribution in the value of a combination of global and local planners (right): arrows pointing upward indicate that the higher the value of a metric the higher the value of the combination; arrows pointing downward indicate that the value of a combination of a global and local planners is higher the lower the value of that metric is

	Type	Proportionality contribution
Global planner evaluation metric		
$\mu_l(\boldsymbol{\mathcal{G}})$	The mean length of the global plans produced $\mu_l(\boldsymbol{\mathcal{G}}) = \mu(l(\boldsymbol{G}_j))$ [m], $j = 1, 2, \ldots, N$	$\searrow$
$\sigma_l(\boldsymbol{\mathcal{G}})$	The standard deviation around the mean global plan length $\sigma_l(\boldsymbol{\mathcal{G}}) = \sigma(l(\boldsymbol{G}_j))$ [m], $j = 1, 2, \ldots, N$, which is a measure of consistency between plans	$\searrow$
$\mu_r(\boldsymbol{\mathcal{G}})$	The mean number of global path poses over the mean global plan length $\mu_r(\boldsymbol{\mathcal{G}}) = \mu(\lvert\boldsymbol{G}_j\rvert / l(\boldsymbol{G}_j))$ [poses / m], $j = 1, 2, \ldots, N$; a measure of the plans' resolution: the higher the resolution, the finer and more delicate the robot's maneuvers (if, for instance, specifications require it)	$\nearrow$
$\mu_s(\boldsymbol{\mathcal{G}})$	The mean smoothness of the produced paths $\mu_s(\boldsymbol{\mathcal{G}}) = \mu(s(\boldsymbol{G}_j))$ [rad], $j = 1, 2, \ldots, N$	$\searrow$
$\sigma_s(\boldsymbol{\mathcal{G}})$	The standard deviation around the mean global plan smoothness $\sigma_s(\boldsymbol{\mathcal{G}}) = \sigma(s(\boldsymbol{G}_j))$ [rad], $j = 1, 2, \ldots, N$	$\searrow$
$\inf(d(\boldsymbol{\mathcal{G}}, \boldsymbol{M}))$	The overall minimum distance of global plans to obstacles in map $\boldsymbol{M}$ across all simulations $\inf(d(\boldsymbol{\mathcal{G}}, \boldsymbol{M}))$ [m]; a measure of how well the global planner plans with respect to obstacle clearance.	$\nearrow$
$\mu(d(\boldsymbol{\mathcal{G}}, \boldsymbol{M}))$	The average mean minimum distance of global plans to obstacles in map $\boldsymbol{M}$, $\mu(d(\boldsymbol{G}_j, \boldsymbol{M}))$ [m], $j = 1, 2, \ldots, N$	$\nearrow$
$\sigma(d(\boldsymbol{\mathcal{G}}, \boldsymbol{M}))$	The standard deviation around the mean minimum distance of global plans to obstacles in map $\boldsymbol{M}$, $\sigma(d(\boldsymbol{G}_j, \boldsymbol{M}))$ [m], $j = 1, 2, \ldots, N$	$\searrow$
Local planner evaluation metric		
μ_A/N	The mean number of aborted missions over the number of simulations conducted	$\searrow$
μ_{RR}	The mean number of rotation recoveries	$\searrow$
σ_{RR}	The standard deviation around the mean number of rotation recoveries	$\searrow$
μ_{CC}	The mean number of costmap clearances	$\searrow$
σ_{CC}	The standard deviation around the mean number of costmap clearances	$\searrow$
μ_{PF}	The mean number of path failures; a measure of how many times the local planner failed to procure valid control inputs	$\searrow$
σ_{PF}	The standard deviation around the mean number of path failures	$\searrow$
μ_{PF}/μ_{LPC}	The mean number of path failures over the mean number of local planner calls; a measure of how often the local planner failed to control the robot's trajectory	$\searrow$
Combination of planners evaluation metric		
$\mu_\delta(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$	The mean deviation between the actual paths $\boldsymbol{\mathcal{P}}$ the robot traversed compared to the global plans $\boldsymbol{\mathcal{G}}$ it was to follow $\mu_\delta(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}}) = \mu(d_\delta(\boldsymbol{P}_j, \boldsymbol{G}_j))$ [m], $j = 1, 2, \ldots, N$	$\searrow$
$\mu_\Delta(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$	The mean total deviation between the actual paths $\boldsymbol{\mathcal{P}}$ the robot took compared to the global plans $\boldsymbol{\mathcal{G}}$ it was to follow $\mu_\Delta(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}}) = \mu(d_\Delta(\boldsymbol{P}_j, \boldsymbol{G}_j))$ [m], $j = 1, 2, \ldots, N$	$\searrow$
$\mu_\delta^F(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$	The mean Frechet distance between the actual paths $\boldsymbol{\mathcal{P}}$ the robot took and their corresponding global paths $\boldsymbol{\mathcal{G}}$, $\mu_\delta^F(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}}) = \mu(\delta_{dF}(\boldsymbol{P}_j, \boldsymbol{G}_j))$ [m], $j = 1, 2, \ldots, N$	$\searrow$
μ_t	The mean travel time from initial pose p_0 to goal pose p_G [sec]	$\searrow$
σ_t	The standard deviation around the mean travel time [sec]	$\searrow$
$\mu_l(\boldsymbol{\mathcal{P}})$	The mean length of the actual paths $\boldsymbol{\mathcal{P}}$ the robot traversed as a result of the local planner's performance $\mu_l(\boldsymbol{\mathcal{P}}) = \mu(l(\boldsymbol{P}_j))$ [m], $j = 1, 2, \ldots, N$	$\searrow$

Table 1 (continued)

	Type	Proportionality contribution
$\sigma_l(\boldsymbol{\mathcal{P}})$	The standard deviation around the mean actual path length $\sigma_l(\boldsymbol{\mathcal{P}}) = \sigma(l(\boldsymbol{P}_j))$ [m], $j = 1, 2, \ldots, N$; a measure of the consistency of the paths the local planner dictated to the robot	$\searrow$
$\mu_s(\boldsymbol{\mathcal{P}})$	The mean smoothness of the traversed paths $\mu_s(\boldsymbol{\mathcal{P}}) = \mu(s(\boldsymbol{P}_j))$ [rad], $j = 1, 2, \ldots, N$	$\searrow$
$\sigma_s(\boldsymbol{\mathcal{P}})$	The standard deviation around the mean traversed paths' smoothness $\sigma_s(\boldsymbol{\mathcal{P}}) = \sigma(s(\boldsymbol{P}_j))$ [rad], $j = 1, 2, \ldots, N$	$\searrow$
$\inf(d(\boldsymbol{\mathcal{P}}, \boldsymbol{M}_C))$	The overall minimum distance of the actual paths the robot traversed to obstacles in $\boldsymbol{M}$ across all simulations $\inf(d(\boldsymbol{\mathcal{P}}, \boldsymbol{M}))$ [m]; a measure of how well the local planner plans the robot's way around obstacles so as not to violate collision-avoidance constraints	$\nearrow$
$\mu(d(\boldsymbol{\mathcal{P}}, \boldsymbol{M}_C))$	The average mean minimum distance of the actual paths the robot traversed to obstacles in map $\boldsymbol{M}$, $\mu(d(\boldsymbol{P}_j, \boldsymbol{M}))$ [m], $j = 1, 2, \ldots, N$	$\nearrow$
$\sigma(d(\boldsymbol{\mathcal{P}}, \boldsymbol{M}_C))$	The standard deviation around the average mean minimum distance of the actual paths the robot traversed in map $\boldsymbol{M}$ $\sigma(d(\boldsymbol{P}_j, \boldsymbol{M}))$ [m], $j = 1, 2, \ldots, N$	

Analogously, for global planners, local planners, or their combinations, if their value regarding m is inversely proportional to the value of m (such as the value of the metric of time taken to get from the initial pose p_0 to the goal pose p_G), the value of C for metric m shall be expressed by $V_{\overline{q}}(C, m)$:

$$V_{\overline{q}}(C, m) = I(C, m) \cdot w_m \cdot (1 - N(m)) \tag{10}$$

Therefore, based on the above, a generic but exact formula for assigning a value $V(C)$ to the performance of each combination of global and local planners C across all aforementioned evaluation metrics and across N simulations in map $\boldsymbol{M}$ is:

$$V_{\boldsymbol{M}}(C) = \sum_m I_Q(m) \cdot V_q(C, m) + I_{\overline{Q}}(m) \cdot V_{\overline{q}}(C, m) \tag{11}$$

where Q denotes the set of metrics whose value is proportional to the value of a combination of global and local planners, and $I_Q(m)$ is the indicator function for m. $V_{\boldsymbol{M}}$, as defined in Eq. 11, is strictly increasing for all values of a metric $m \in [\min m, \max m]$ on a given map, i.e. for all values of metrics that result from either successful or unsuccessful combinations of planners in that map (as mentioned before, successful in the sense of completing all missions).

The final overall ranking of the performance of all combinations of global and local planners in a map $\boldsymbol{M}$ will therefore be the result of a sort operation on the values of $V_{\boldsymbol{M}}(C)$, as given in Eq. 11, in descending order. The final overall ranking of the performance of all combinations across different maps will be the result of a sort operation on the sum of values of $V_{\boldsymbol{M}}$ across maps $\boldsymbol{M}$.

4.4 Software-Oriented Qualitative Metrics

Along with the quantitative evaluation metrics mentioned above we shall evaluate the global and local planners' status of their software form − their ROS package quality with respect to the metrics below:

- Their documentation quality and wealth; a quality much sought-after by robotics engineers or programmers, regardless of whether their work is mostly focused on theory or practice.
- Their up-to-dateness; a compound quality summing (a) the package's relevancy with respect to its peers, (b) the support offered by its maintainers, (c) the package's maintenance status and (d) its ability to be installed on a robot, i.e. its contemporaneousness with core software such as the robot's operating system, compiler, libraries and, in general, dependencies such as these.
- Their ease of installation.
- Their self-containment/completeness, i.e. no additional piece of software or care is required for the package to operate as intended.
- Their computation needs.
- Their parameterisability; although the number of package parameters increases the complexity of tuning, the ability to (a) adapt the planners' performance to specific instantiations of robots' properties (their geometry in space, their kinematic model, etc), along with (b) tune their parameters in finer detail, ergo their performance (and ultimately the robot's behaviour) under various and variable specifications is paramount to achieving the desired/prescribed task or motion performance by the robot. This quality is coupled with the one above: a wealth of parameters to tune is

irrelevant if no or inadequate illustration about their identity/effect on the whole is made.
- Their consistency in their performance, i.e. the failures they exhibit due to inadequate translation of the theory behind the software implementation into programming code (this includes (in-)consistency in emergence of software bugs, and speed of execution).

Before we move on to the evaluation of all planners' combinations we shall sift through them based on the first five qualitative metrics defined above: an obsolete, non-self-contained, or misinstallable package is a package that cannot be used in practice; a package lacking in documentation is a package which, even if it's usable, it deprives the engineer of insight into its method, obstructs access to or obscures the meaning of its parameters and therefore hinders its parameterisability and, ultimately, its usability and longevity; finally, a resource-hungry package is a package denying other nodes resources they need, therefore compromise their performance and the performance of the joint robotic system.

5 Global-Local Planners Evaluation

5.1 Initial Sifting

Packages `navfn` and `global_planner` are considered to be the default choices for global planners in ROS: they are the oldest and (considered to be−) the a priori safest choices for navigation. Moreover, they require little (if not no) tuning at all. They have been continuously maintained since the early days of ROS (this applies especially for `navfn`) and, as the de facto ROS global planners to be used, they will be regarded as a baseline for all other global planners that pass through the initial sifting phase.

We consider package `asr_navfn` to be redundant as (a) its behaviour is exactly the same as `navfn`'s and (b) its employability rests on the potential failure of the robot's target selector. Furthermore, it is not currently maintained (the latest ROS-version supported is `kinetic` and the latest `gihub` commit is two years old at the time of writing of this article). Therefore this package will not be evaluated in accordance with the second qualitative criterion.

Although package `MoveIt!` is heavily documented, supported, and up-to-date, it is not targeted at $2D$ navigation. Therefore this package will not be evaluated, in accordance with the second qualitative criterion. In Table 2, `MoveIt!` receives three bullets in the column for computation needs due to its need for resources in order to plan for multi-degree of freedom robotic arms with various and varying spatio-temporal complicated constraints.

Package `sbpl_lattice_planner` is documented both in theory and parametrically. It is currently up-to-date with the latest ROS version (currently `melodic`), maintained, and supported by ROS maintainers (a software bug discovered during its evaluation was squashed within 8 days). Its installation (apart from that of its core library `SBPL`) required virtually no effort.

Table 2 Qualitative metrics evaluation of (as defined in Section 4.4) and acceptance decision for inclusion in evaluation for all planners considered

	Qualitative metrics							
Planner	DOC	UTD	INST	SC/C	PARAM	CON	COMP	Accepted
`navfn`	●	●	●●	●	●	●	●	●
`global_planner`	●	●	●●	●	●	●	●	●
`asr_navfn`	●	○	●	●	●	●	●	○
`MoveIt!`	●●●	●	●●	●	●●●	?	●●●	○
`sbpl_lattice_planner`	●●	●	●●	●	●	○	●	●
`sbpl_dynamic_env_global_planner`	●	○	●	○	●	?	●	○
`lattice_planner`	●	○	●	●	●	●	●	○
`waypoint_global_planner`	●	○	●	○	○	●	●	○
`voronoi_planner`	●	○	●	●	●	●	●	○
`dwa_local_planner`	●	●	●●	●	●	●	●	●
`eband_local_planner`	●	○	●●	●	●●	●	●●	●
`teb_local_planner`	●●●	●	●●	●	●●●	●●	●●	●

Abbreviations are introduced to economize on space; DOC is an abbreviation of a package's documentation quality, UTD of its up-to-dateness, INST of its ease of installation, SC/C of its self-containment/completeness, PARAM of its parameterisability, CON of its consistency in execution, and COMP of its computation needs. Hollow bullets indicate the inadequacy of planners with respect to each metric. Question marks indicate unknown status

The dynamic version of the sbpl_lattice_planner, package sbpl_dynamic_env_global_planner is considered to be redundant since this article is dealing with navigation in static environments; nevertheless its reference page warns the reader that the tracker used to track moving objects is not robust (especially when the robot is on the move), advising her to provide a better alternative. Furthermore, it requires the substitution of the whole move_base package with a modification of it, so that both global and local planners run at the same time in parallel. Finally, it is evaluated as not up-to-date, since the latest confirmed ROS distribution is diamondback, and its latest update was over 5 years ago at the time of writing of this article. Ergo this package is neither self-contained nor up-to-date, and, in accordance with criteria two and four, will not be considered in the coming evaluation.

Although package lattice_planner is documented and self-contained, it is not actively maintained (its latest github commit is three years old) and, therefore will not be evaluated, in accordance with the second qualitative criterion.

The same applies for package waypoint_global_planner: it is minimally documented, not actively maintained, and not self-contained in the sense that providing the initial pose p_0 and a goal pose p_G is not sufficient for the generation of a path connecting p_0 to p_G since the planner is unable to consider obstacles in the global costmap. Ergo it will not be considered for evaluation, in accordance to the first, second and fourth criteria.

As for package voronoi_planner, it is also minimally documented and not actively maintained (the latest ROS-version supported is indigo and the latest github commit is three years old) and, therefore, will not be considered for evaluation, in accordance with the first and second criteria.

Regarding the local planners, the status of dwa_local_planner is equivalent to that of navfn and global_planner: it is the baseline (local) planner in ROS.

Local planner eband_local_planner is documented, installed through the standard package installation procedure, and self-contained. However, it has not been updated to match the latest ROS versions[24], nor does it seem to be currently or actively maintained. Nevertheless, we shall include it in our local planners evaluation as an exception due to the critical lack of local planners in ROS. In Table 2, under the column for computation needs, eband_local_planner receives two bullets due to its need of solving a non-linear constrained optimisation problem online.

[24] https://github.com/utexas-bwi/eband_local_planner/issues/28

Lastly, local planner teb_local_planner can be said to be the most well- and thoroughly-documented planner among all global and local ones, on both theoretical and parametric levels. It is up-to-date with the latest ROS releases, self-contained, and it is the most parameterisable planner in the ROS ecosystem. Just as eband_local_planner, teb_local_planner receives two bullets in the column for computation needs in Table 2, due to its need of solving a non-linear spatio-temporally constrained optimisation problem online.

Overall, none of the planners discussed above are excessively resource-hungry, and therefore their employment and operation beside the other packages (the localisation or SLAM module for instance) does not compromise the latter's performance.

Table 2 illustrates the complete evaluation list based on qualitative criteria of Section 4.4 for all planners considered.

Table 3 features the planners that will be considered for evaluation. The notation GP and LP used hereafter in the header of tables is shorthand for "Global Planners" and "Local Planners" respectively.

5.2 Evaluation on Map CORRIDOR

Overall, all combinations of dwa_local_planner with any global planner failed in making the robot reach p_G^C, and the same is observed for the combination of sbpl_lattice_planner with eband_local_planner. The remaining combinations were reliable each time run. Table 4 summarises the success rate of each combination of global and local planners in map $\boldsymbol{M}_C$.

Figure 5 depicts the global plans produced by all global planners featured in the first column of Table 3 for all combinations of global and local planners of the same table, and over N simulations for each combination, in regard to navigation in map CORRIDOR $\boldsymbol{M}_C$.

Figure 6 depicts the actual paths traversed by the robot for all combinations of global and local planners of the same table, and over N simulations for each combination.

Table 4 features the value $V_{\boldsymbol{M}_C}$ and rank of all combinations of global and local planners evaluated on all metrics exhibited in Tables 8, 9, 10, 11, 12, 13

Table 3 The list of global and local planners' ROS packages whose combination's performance will be evaluated

Global planners (GP)	Local planners (LP)
navfn	dwa_local_planner
global_planner	eband_local_planner
sbpl_lattice_planner	teb_local_planner

Table 4 The success rate, value $V_{\boldsymbol{M}_C}$, and corresponding rank of all combinations of global and local planners evaluated on their performance in the map of world CORRIDOR $\boldsymbol{M}_C$ across $N = 10$ simulations

GP	LP	Successful missions / N	$V_{\boldsymbol{M}_C}$	rank
`navfn`	`teb`	10/10	21.41	1
`sbpl`	`teb`	10/10	20.35	2
`globalplanner`	`teb`	10/10	19.29	3
`navfn`	`eband`	10/10	15.96	4
`globalplanner`	`eband`	10/10	14.70	5
`sbpl`	`eband`	0/10	10.99	6
`sbpl`	`dwa`	0/10	6.56	7
`navfn`	`dwa`	0/10	6.46	8
`globalplanner`	`dwa`	0/10	5.50	9

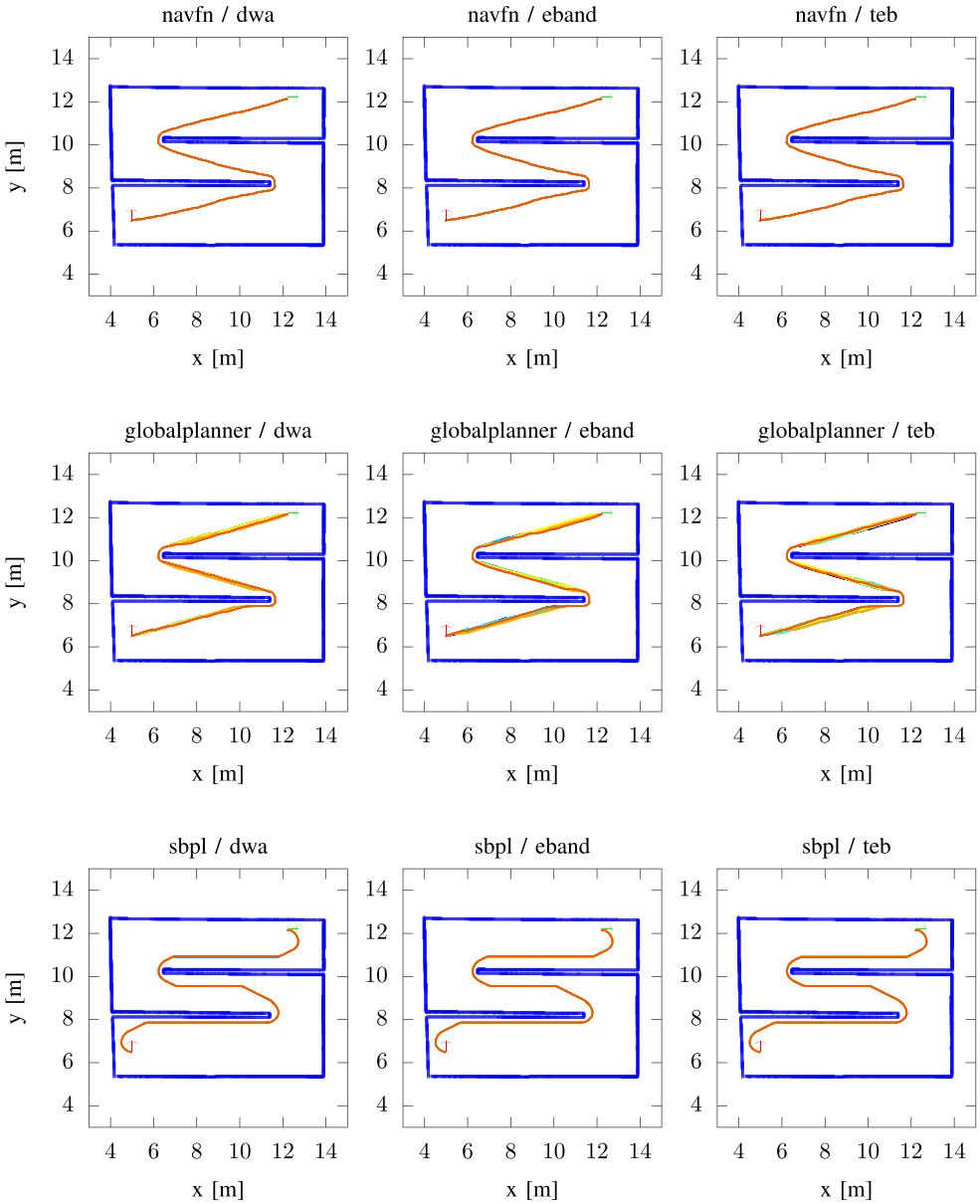

Fig. 5 Global plans $\mathcal{G}$ produced by the three global planners with regard to the initial and goal poses of map CORRIDOR $\boldsymbol{M}_C$. Each row depicts the global plans produced by one global planner when combined with each of the reviewed local planners. Navigation with each combination of planners was run $N = 10$ times

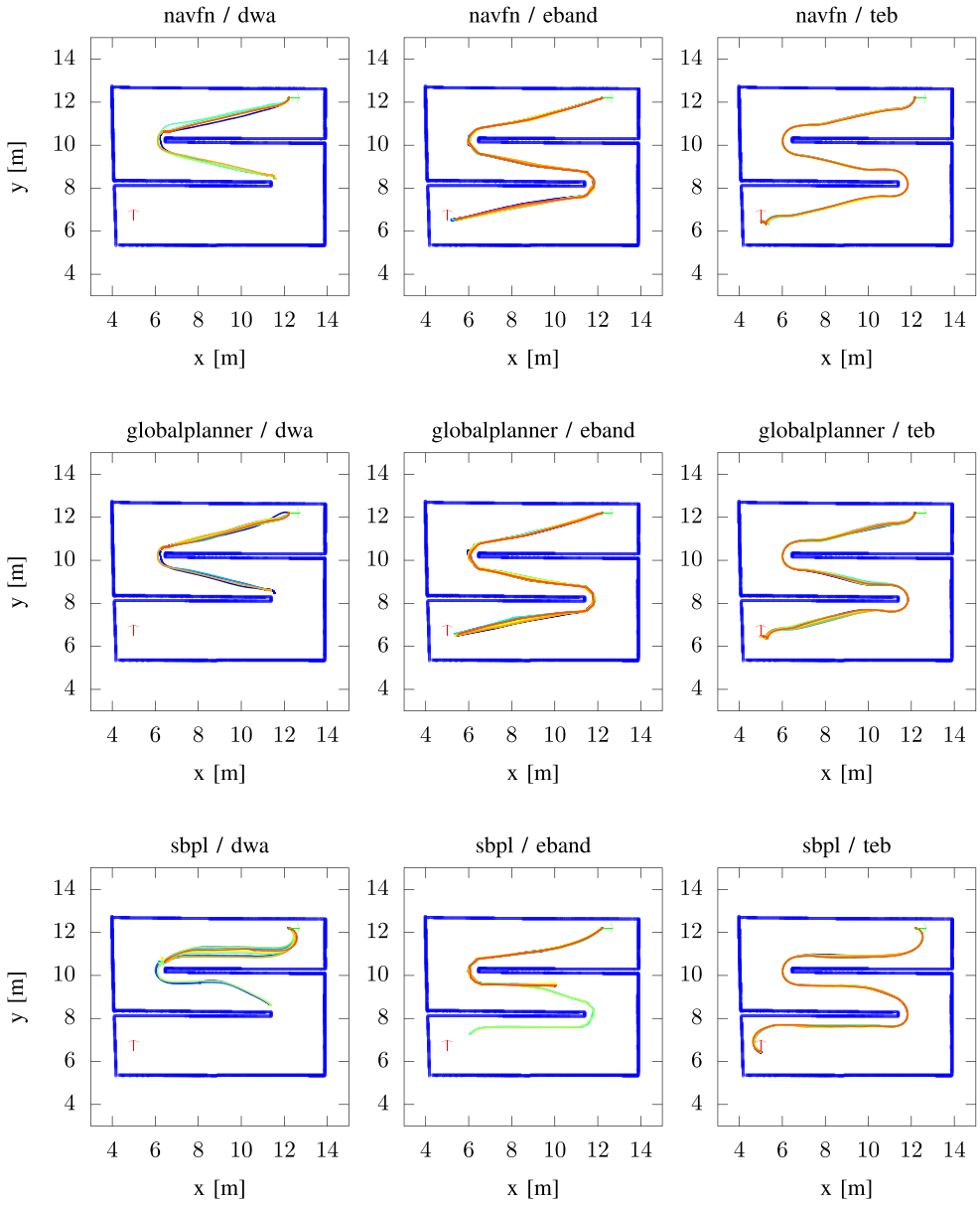

Fig. 6 The paths traversed $\mathcal{P}$ by the robot with regard to the initial and goal poses of map CORRIDOR $\boldsymbol{M}_C$. Each column depicts the paths that were the result of application of one local planner when combined with each of the reviewed global planners. Navigation with each combination of planners was run $N = 10$ times

Table 5 The success rate, value V_{M_W} and corresponding rank of all combinations of global and local planners evaluated on their performance in the map of world WILLOWGARAGE M_W across $N = 10$ simulations

GP	LP	Successful missions / N	$V_{\boldsymbol{M}_W}$	rank
`globalplanner`	`teb`	10/10	21.90	1
`navfn`	`teb`	10/10	20.00	2
`sbpl`	`teb`	10/10	12.27	3
`globalplanner`	`eband`	0/10	11.95	4
`navfn`	`eband`	0/10	11.76	5
`sbpl`	`eband`	0/10	9.85	6
`navfn`	`dwa`	0/10	9.31	7
`globalplanner`	`dwa`	0/10	8.86	8
`sbpl`	`dwa`	0/10	4.85	9

and 14 regarding their performance in navigation on the map of world CORRIDOR. For the calculation of the value of all combinations, all weights $w_m = 1.0$ except that corresponding to the local-planner-specific metric of μ_{PF}/μ_{LPC}, due to the fact that `eband_local_planner` does not provide access to the number of times it was called to direct the motion of the robot, which was set to 0.0. Overall, local planner `teb_local_planner` occupied all podium places, with its combination with global planner `navfn` being the best-performing among the three. Details on the performance of global planners, local planners, and their combination, resides in Appendix B.

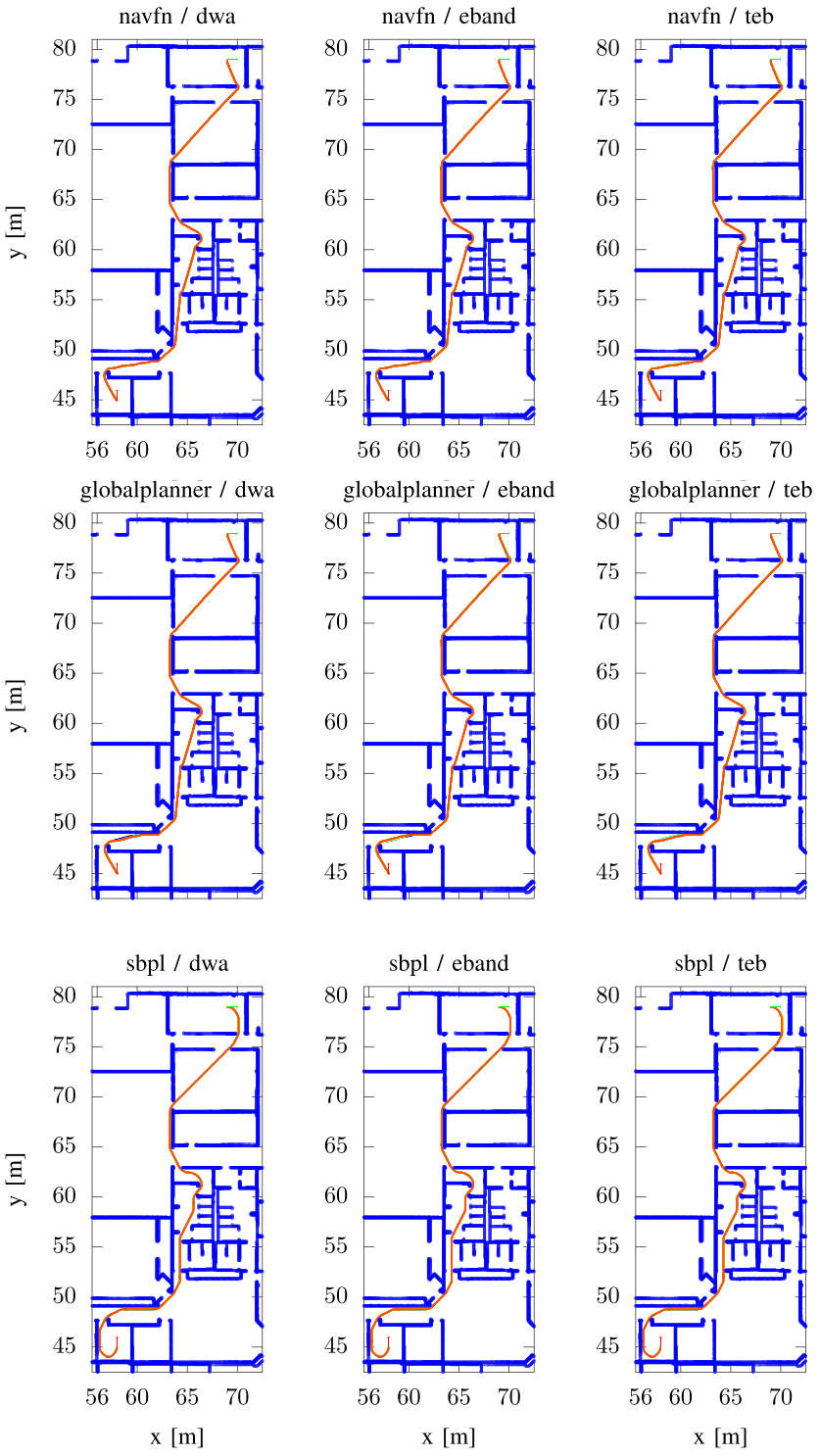

Fig. 7 Global plans $\mathcal{G}$ produced by the three global planners with regard to the initial and goal poses of map WILLOWGARAGE $\boldsymbol{M}_W$. Each row depicts the global plans produced by one global planner when combined with each of the reviewed local planners. Navigation with each combination of planners was run $N = 10$ times

5.3 Evaluation on map WILLOWGARAGE

Overall, the combinations of `dwa_local_planner` and `eband_local_planner` with all global planners failed in making the robot reach p_G^W. The remaining combinations (all having `teb_local_planner` as their local planner) were reliable each time run. Table 5 summarises the success rate of each combination of global and local planners in map $\boldsymbol{M}_W$.

Figure 7 depicts the global plans produced by all global planners featured in the first column of Table 3 for all combinations of global and local planners of the same table, and over N simulations for each combination, in regard to navigation in map WILLOWGARAGE $\boldsymbol{M}_W$.

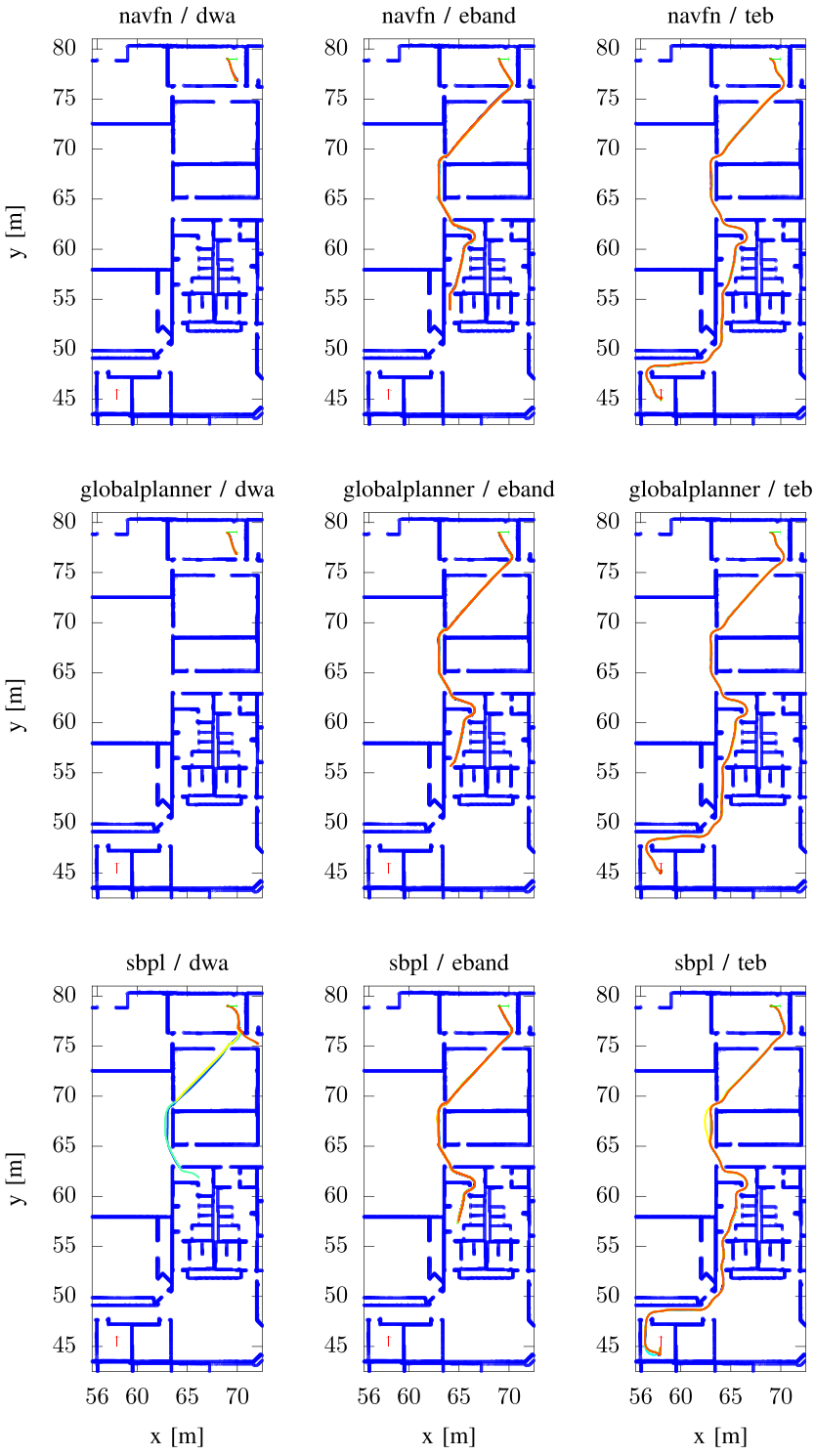

Fig. 8 The paths traversed $\mathcal{P}$ by the robot with regard to the initial and goal poses of map WILLOWGARAGE $\boldsymbol{M}_W$. Each column depicts the paths that were the result of application of one local planner when combined with each of the reviewed global planners. Navigation with each combination of planners was run $N = 10$ times

Table 6 The success rate, value $V_{\boldsymbol{M}_L}$ and corresponding rank of all combinations of global and local planners evaluated on their performance in the map of world CSAL $\boldsymbol{M}_L$ across $N = 10$ experiments

GP	LP	Successful missions / N	$V_{\boldsymbol{M}_C}$	rank
navfn	teb	10/10	18.74	1
globalplanner	teb	10/10	16.84	2
navfn	eband	10/10	14.77	3
globalplanner	eband	10/10	14.26	4
sbpl	teb	10/10	13.57	5
navfn	dwa	0/10	8.10	6
sbpl	eband	10/10	7.80	7
sbpl	dwa	0/10	6.47	8
globalplanner	dwa	0/10	6.13	9

Figure 8 depicts the actual paths traversed by the robot for all combinations of global and local planners of the same table and over N simulations for each combination.

Table 5 features the value $V_{\boldsymbol{M}_W}$ and rank of all combinations of global and local planners evaluated on all metrics exhibited in Tables 15, 16, 17, 18, 19, 20 and 21, regarding their performance in navigation on the map of world WILLOWGARAGE. For the calculation of the value of all combinations, all weights $w_m = 1.0$ except that corresponding to the local-planner-specific metric of

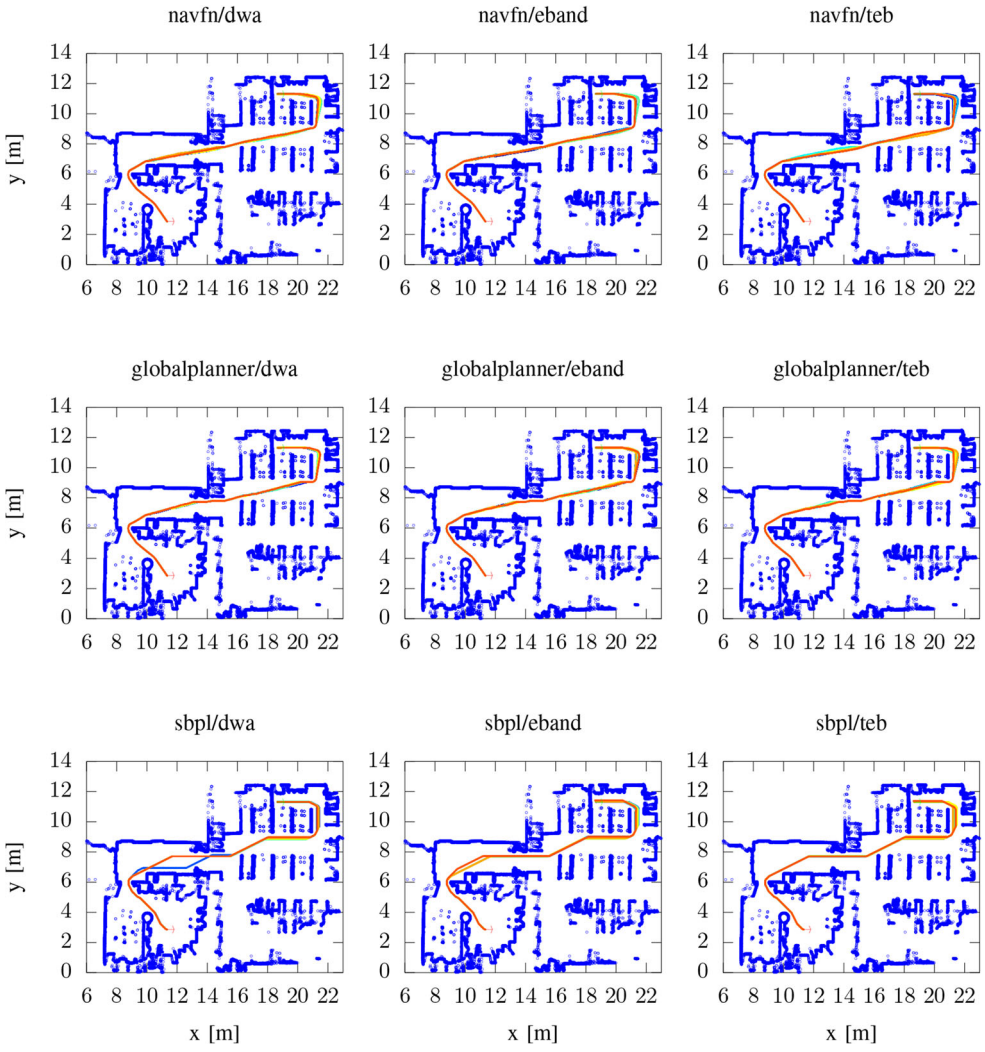

Fig. 9 Global plans $\mathcal{G}$ produced by the three global planners with regard to the initial and goal poses of map CSAL $\boldsymbol{M}_L$. Each row depicts the global plans produced by one global planner when combined with each of the reviewed local planners. Navigation with each combination of planners was run $N = 10$ times

μ_{PF}/μ_{LPC}, which was again set to 0. Local planner `teb_local_planner` again occupied all podium places (this time by virtue of all other local planners failing to complete the robot's mission to travel from the initial to the goal pose), with its combination of `global_planner` outperforming that with `navfn`, which was the overall best in the map of world CORRIDOR. Details on the performance of global planners, local planners, and their combination, resides in Appendix C.

5.4 Evaluation on Map CSAL

Overall, as in simulations, all combinations of `dwa_local_planner` with any global planner failed in making the robot reach p_G^L. The remaining combinations were reliable each time run. Table 6 summarises the success rate of each combination of global and local planners in map $\boldsymbol{M}_L$.

Figure 9 depicts the global plans produced by all global planners featured in the first column of Table 3 for all combinations of global and local planners of the same table, and over N simulations for each combination, in regard to navigation in map CSAL $\boldsymbol{M}_L$.

Figure 10 depicts the actual paths traversed by the robot for all combinations of global and local planners of the same table, and over N simulations for each combination.

Table 6 features the value $V_{\boldsymbol{M}_L}$ and rank of all combinations of global and local planners evaluated on all metrics exhibited in Tables 22, 23, 24, 25, 26, 27 and 28 regarding their performance in navigation on the map of environment CSAL. For the calculation of the value of all combinations, all weights $w_m = 1.0$ except that corresponding to the local-planner-specific metric of μ_{PF}/μ_{LPC}, due to the fact that `eband_local_planner` does not provide access to the number of times it was called to direct the motion of the robot, which was set to 0.0.

What stands out in real-life experiments is that the performance of combinations of global planner `sbpl_lattice_planner` with local planners decreased, allowing `eband_local_planner` – although slower, as

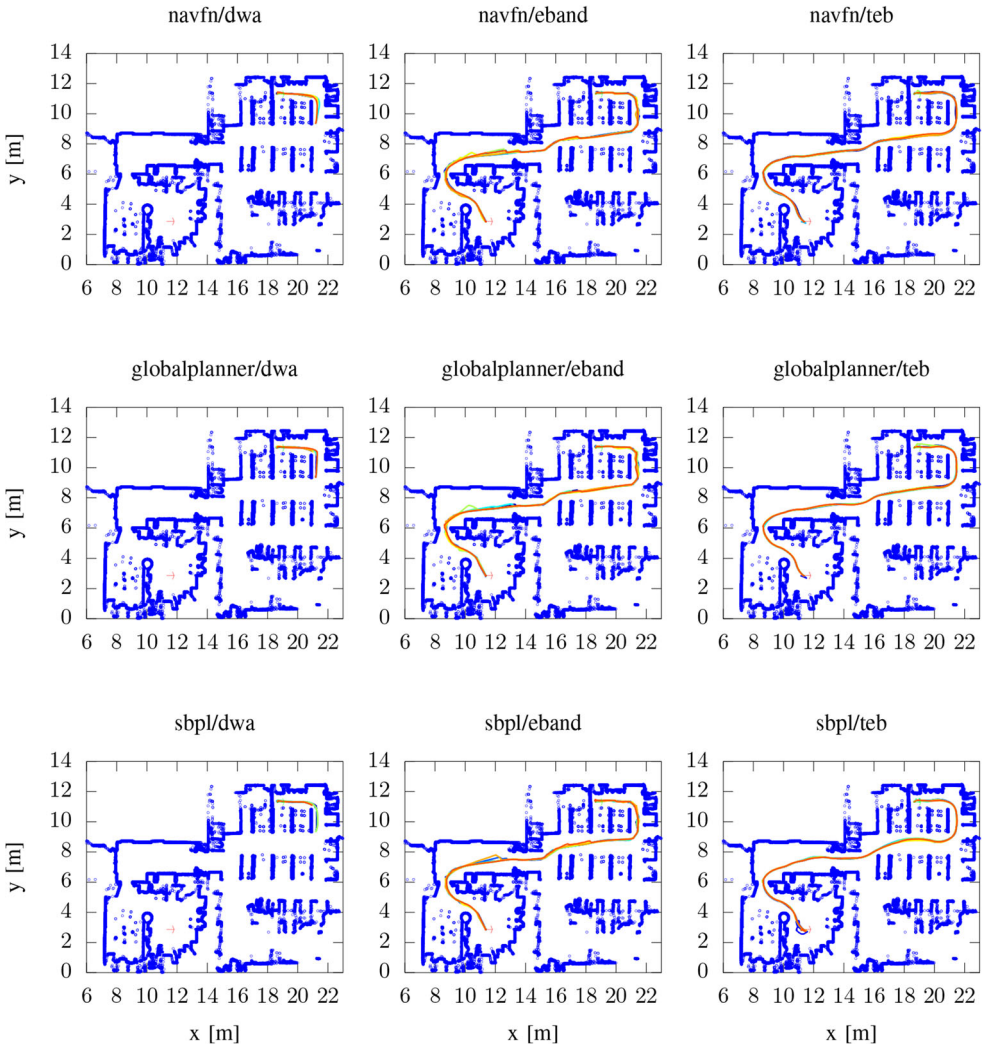

Fig. 10 The paths traversed $\mathcal{P}$ by the robot with regard to the initial and goal poses of map CSAL $\boldsymbol{M}_L$. Each column depicts the paths that were the result of application of one local planner when combined with each of the reviewed global planners. Navigation with each combination of planners was run $N = 10$ times

Table 7 The compound value V for and corresponding rank of all combinations of global and local planners evaluated on their performance in the maps of worlds CORRIDOR $\boldsymbol{M}_C$, WILLOWGARAGE $\boldsymbol{M}_W$, and CSAL $\boldsymbol{M}_L$

GP	LP	$V_{\boldsymbol{M}_C}$	$V_{\boldsymbol{M}_W}$	$V_{\boldsymbol{M}_L}$	V	rank
navfn	teb	21.41	20.00	18.74	60.15	1
globalplanner	teb	19.29	21.90	16.84	58.03	2
sbpl	teb	20.35	12.27	13.57	46.19	3
navfn	eband	15.96	11.76	14.77	42.49	4
globalplanner	eband	14.70	11.95	14.26	40.91	5
sbpl	eband	10.99	9.85	7.80	28.94	6
navfn	dwa	6.46	9.31	8.10	28.64	7
globalplanner	dwa	5.50	8.86	6.13	20.49	8
sbpl	dwa	6.56	4.85	6.47	17.88	9

in simulations – and its combinations with the remaining two global planners to displace teb_local_planner's combination with sbpl_lattice_planner from the top positions. Setting the combinations of this global planner with local planners aside, the combinations of the remaining global planners exhibit the same pattern observed in simulations: (a) given a global planner, teb_local_planner outperforms eband_local_planner, which in turns outperforms dwa_local_planner, and (b) given a local planner, navfn outperforms global_planner.

Details on the performance of global planners, local planners, and their combination, resides in Appendix D.

5.5 Overall Evaluation

Table 7 features the values of all combinations of global planners of Table 3 across maps of environments CORRIDOR, WILLOWGARAGE, and CSAL.

Two definitive patterns emerge clearly: as regards local planners, teb_local_planner outperforms eband_local_planner, which outperforms dwa_local_planner; as regards global planners, given a local planner, navfn outperforms global_planner by a small margin, smaller than that between the latter and sbpl_lattice_planner, and this applies for all combinations of global planners with all local planners. Therefore, it is reasonable to assume that the best combination of planners for use on autonomous navigation of ground vehicles in 2D occupancy-grid maps employs navfn as its global planner component and teb_local_planner as its local planner component. Furthermore, the best candidates for substitution of the above planners, depending on the conditions of the environment and the setup of the robot, are global planner global_planner and local planner eband_local_planner.

6 Conclusions

This work evaluates the state-of-the-art global and local planners whose source code is available for direct deployment via the Robot Operating System (ROS) on the task of navigation of Unmanned Ground Vehicles – specifically in this work a Turtlebot v2 – in 2D occupancy grid maps. An initial sift was performed in order to distinguish robust planners based on software-oriented qualitative metrics. The planners that passed through this sifter were combined so that the performance of all combinations of global and local planners would be evaluated. Their performance was tested on two heterogeneous simulated environments and on one real-world environment, so that both components would face appropriate stress tests.

Overall, we discern the local planner teb_local_planner as the most robust and definitively successful local planner among the ones tested: it never failed to navigate from an initial pose to a goal one, and it always did so in the least amount of time. Local planner eband_local_planner came in second, failing to navigate in an appropriate amount of time in one simulated environment map, while clearly directing the motion of the robot in an overly-safe manner. However, in real-world experiments, it performed better than in simulations. The third local planner, dwa_local_planner did not successfully complete a single mission in either environment, simulated or real.

As regards global planners, navfn and global_planner are almost equivalent, with the former outperforming the latter overall. On the other hand, sbpl_lattice_planner's gap of performance between global_planner is greater than that between the latter and navfn.

European Union
European Regional Development Fund

EPAnEK 2014-2020
OPERATIONAL PROGRAMME
COMPETITIVENESS ENTREPRENEURSHIP INNOVATION

Co-financed by the European Union and Greek national funds

ΕΣΠΑ 2014-2020
ανάπτυξη - εργασία - αλληλεγγύη
Partnership Agreement 2014-2020

Acknowledgments This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH CREATE INNOVATE (project code:T1EDK-03032).

The authors would like to thank the reviewers of this paper for their helpful comments and suggestions.

Appendix A: Proportionality Contribution of Metrics to the Value of a Combination of Planners

On the value of a combination of global and local planners that pertains to metrics of its **global planner component**, we make the following assumptions. The value of a combination of global and local planners is: (a) higher the shorter a global plan is − so that a robot traversing it in constant speed takes less time to get from an initial to a goal pose, (b) higher the higher its resolution is − the finer a plan's detail the more probable it is that a (sub)goal exists within the horizon of the local costmap, and the more smooth the path may be, (c) higher the more smooth it is − the smoother the plan the more probable it is that the robot takes less time to traverse the path from the initial pose to the goal one (following the global plan to the letter is a matter of the local planner as regards how fit it considers the global plan to be, and how feasible it actually is) (d) higher the larger its average mean minimum distance from obstacles in a map is − so that collisions with obstacles are less probable to occur (one must count on the uncertainty of improbable events) (e) higher the larger its overall minimum distance from obstacles in a map is across all simulations, and (f) lower the more varied the value of each metric is − so that the engineer can count on its reliability. As is evident, these metrics are independent of the success or failure of combinations of global and local planners in reaching the goal pose p_G from the initial one p_0, i.e. they are included in a combination's calculation of value regardless if that combination failed to complete all missions.

On the value of a combination of global and local planners that pertains to metrics of its **local planner component**, we make the following assumptions. The value of a combination of global and local planners is: (a) lower the higher the mean number of aborted missions across the sum of its performed simulations is, (b) lower the higher the mean number of rotation recoveries performed is, (c) lower the higher the mean number of costmap clearances performed is, (d) lower the higher mean number of path failures exhibited is, (e) lower the higher the relative number of path failures is, and (f) lower the more varied the value of each metric is. Again, these metrics are independent of the success or failure of combinations of global and local planners in reaching the goal pose, and will therefore be included in a combination's calculation of value regardless if that combination failed every single mission or not.

On the value of a combination of global and local planners that pertains to metrics referring to their **joint performance**, we make the following assumptions. The value of a combination of global and local planners is: (a) lower the larger the mean deviation of the actual paths the robot took − as a result of the application of the local planner − from the paths the global planner designed the robot to take, (b) lower the larger the mean total deviation of the former from the latter is, (c) lower the larger the mean Frechet distance of the former from the latter is, (d) higher the lower the travel time from the initial to the goal pose is, (e) higher the shorter the actual paths the robot took are, (f) lower the less smooth the actual paths the robot took are, (g) higher the larger the robot's average mean minimum distance from obstacles in a map is, (e) higher the larger the robot's overall minimum distance from obstacles in a map is across all simulations, and (f) lower the more varied the value of each metric is. The above metrics are dependent on the success or failure of the combination of global and local planners in reaching the goal pose, and will therefore not be included in the calculation of the value of a combination if that combination failed to reach p_G for every simulation that this occurred.

Appendix B: Details of Evaluation on Map CORRIDOR

1) Global-planner-specific evaluation

Tables 8 and 9 illustrate the resulted values of the quantitative metrics concerning global planners defined in Table 1 over $N = 10$ simulations in environment CORRIDOR.

Regarding the produced plans, `global_planner` produced paths of the least length (Table 8), `sbpl_lattice_planner` those of the most length and of the lowest density but of the highest smoothness (lower numbers indicate higher smoothness), and `navfn` produced the least

Table 8 Mean global plan length $\mu_l(\mathcal{G})$ and standard deviation $\sigma_l(\mathcal{G})$, mean global plan resolution $\mu_r(\mathcal{G})$, and mean $\mu_s(\mathcal{G})$ and standard deviation $\sigma_s(\mathcal{G})$ of the global plans' smoothness for map CORRIDOR $\boldsymbol{M}_C$

		Global-planner-specific metrics				
GP	LP	$\mu_l(\mathcal{G})$ [m]	$\sigma_l(\mathcal{G})$ [m]	$\mu_r(\mathcal{G})$ [poses/m]	$\mu_s(\mathcal{G})$ [rad]	$\sigma_s(\mathcal{G})$ [rad]
navfn	dwa	19.63	0.00	76.18	2.42	0.00
navfn	eband	19.63	0.00	76.18	2.42	0.00
navfn	teb	19.61	0.02	76.20	2.42	0.00
globalplanner	dwa	19.60	0.01	74.43	2.40	0.00
globalplanner	eband	19.59	0.01	74.70	2.40	0.00
globalplanner	teb	19.60	0.01	74.70	2.40	0.00
sbpl	dwa	22.92	0.00	53.25	2.39	0.00
sbpl	eband	22.92	0.00	53.41	2.39	0.00
sbpl	teb	22.92	0.00	53.33	2.39	0.00

coarse plans but those of the lowest smoothness. The middle's performance with respect to length is reasonable since `sbpl_lattice_planner` *does* take account of the robot's kinematic model, which, being a differential drive robot and therefore non-holonomic, is constrained in its motion. In contrast, `navfn` and `global_planner` do not consider such constraints and, the latter being the successor of the former, produce slightly similar plans (this is also observed when considering the figures of the two planners' produced paths: they seem almost identical to the naked eye, in stark contrast to those of `sbpl_lattice_planner`). Another observable difference in Fig. 5 is that the plans of `navfn` and most of those of `sbpl_lattice_planner` are deterministic: given an initial pose p_0, a goal pose p_G and a map, they produce the same plan each time, while `global_planner` introduces a slight degree of randomness, which explains why its plans' standard deviation is non-zero compared to the other two planners.

With regard to the crucial ability of a global planner to plan around obstacles (Table 9), `global_planner` produced paths that did not fully take account of the provided robot radius (subtracting the robot's radius from the overall minimum distance of its plans to the closest obstacle gives −0.02m), and therefore (a) a local planner of full fidelity to the global plan would, with certainty, force the robot to halt its mission (until perhaps a new target was set), or even to collide with obstacles in its environment and (b) the engineer should increase `move_base`'s view of the robot's radius in its costmap parameters. The two remaining global planners produced paths that graze obstacles at least once. Furthermore, `sbpl_lattice_planner` would like to the robot to move parallel to walls, which is a behaviour that can actually be dictated to the planner (the planner was tuned so that the robot would prefer to move in straight lines), which can be considered somewhat advantageous since there is always an obstacle close enough so that it can be exploited as a frame of reference under the task of either mapping or localisation.

2) Local-planner-specific evaluation

Table 10 summarises the resulted values of the quantitative metrics concerning local planners defined in Table 1 over N simulations.

Table 9 Overall minimum distance of the global plans from any obstacle across all experiments $\inf(d(\mathcal{G}, \boldsymbol{M}_C))$, mean minimum distance $\mu(d(\mathcal{G}, \boldsymbol{M}_C))$ and standard deviation $\sigma(d(\mathcal{G}, \boldsymbol{M}_C))$ from all obstacles for map CORRIDOR $\boldsymbol{M}_C$

		Global-planner-obstacle-specific metrics		
GP	LP	$\inf(d(\mathcal{G}, \boldsymbol{M}_C))$ [m]	$\mu(d(\mathcal{G}, \boldsymbol{M}_C))$ [m]	$\sigma(d(\mathcal{G}, \boldsymbol{M}_C))$ [m]
navfn	dwa	0.00	0.52	0.32
navfn	eband	0.00	0.52	0.32
navfn	teb	0.00	0.52	0.32
globalplanner	dwa	0.00 (−0.02)	0.48	0.31
globalplanner	eband	0.00 (−0.02)	0.48	0.31
globalplanner	teb	0.00 (−0.02)	0.48	0.32
sbpl	dwa	0.00	0.29	0.20
sbpl	eband	0.00	0.29	0.20
sbpl	teb	0.00	0.29	0.20

Table 10 Mean number of aborted missions over the number of simulations conducted μ_A/N, mean number of rotation recoveries μ_{RR} and their standard deviation σ_{RR}, mean number of costmap clearances μ_{CC} and their standard deviation σ_{CC}, mean number of path failures μ_{PF} and their standard deviation σ_{PF}, and mean number of path failures over the mean number of local planner calls μ_{PF}/μ_{LPC} for all combinations of global and local planners featured in Table 3 on map CORRIDOR $\boldsymbol{M}_C$

		Local-planner-specific metrics							
GP	LP	μ_A/N	μ_{RR}	σ_{RR}	μ_{CC}	σ_{CC}	μ_{PF}	σ_{PF}	μ_{PF}/μ_{LPC}
navfn	dwa	0.90	2.90	0.57	3.30	0.67	53.50	17.35	0.11
globalplanner	dwa	0.90	3.30	1.16	2.70	0.95	58.90	22.29	0.10
sbpl	dwa	0.50	3.30	0.67	3.00	1.41	8.50	5.58	0.02
navfn	eband	0.00	0.00	0.00	0.00	0.00	1.10	1.66	N/A
globalplanner	eband	0.00	0.00	0.00	0.00	0.00	1.60	1.84	N/A
sbpl	eband	0.00	0.00	0.00	0.00	0.00	0.20	0.42	N/A
navfn	teb	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
globalplanner	teb	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sbpl	teb	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

All combinations of local planner `dwa_local_planner` with global planners did not reach the goal pose even once, and this is due to the fact that it spent most of its time executing recovery behaviours (it has the highest mean rotation recoveries and costmap clearances among the three local planners), which resulted either in aborting the missions or in timing-out (ultimately, this is due to the fact that `dwa_local_planner` follows global plans with high fidelity, plans which are actually infeasible, since the overall minimum distance from obstacles is at most zero – Table 9). What is more is that it features the highest ratio of path failures (roughly one out of every ten times the local planner was invoked when the global planner used did not account for the robot's kinematics, resulting in the former not being able to find valid motor inputs).

Local planner `eband_local_planner` performed better than `dwa_local_planner`: it never aborted a mission, it did not execute a single recovery behaviour, and it failed to procure valid motor commands with vastly lower frequency (although this planner does not offer access to its produced plans, and therefore there is no way of knowing how many times it was called to procure them, it is reasonable to assume that, given its low number of mean path failures, its failure frequency follows from the magnitude of the mean path failures, which was lower than that of `dwa_local_planner`). Its ultimate failure is that it is not fast enough (this can be observed in the mean travel times featured in Table 11; recount that $t_C^{max} = 120$ sec), i.e. its approach is overly safe. Its collaboration with global planner `sbpl_lattice_planner` was the worst, which could in theory be partially attributed to the fact that the latter produces the coarsest and lengthiest plans, but is in fact due to an unknown issue that causes the local planner to declare that the robot reached its goal while in fact still mid-way for some simulations (this is the second reason why `sbpl_lattice_planner` was given an inadequacy status in Table 2 – the first being the bug found and resolved that was mentioned in Section 5.1).

In contrast, local planner `teb_local_planner` championed in every available local-planner-specific metric: it never aborted a mission, it did not execute a single recovery behaviour, it never failed to procure valid motor inputs, and it never failed the robot in reaching the goal pose p_G.

3) Global/local-planners-combination-specific evaluation

Tables 11, 12, 13, and 14 summarise the resulted values of the quantitative metrics concerning the combination of global and local planners defined in Table 1 over N

Table 11 Mean travel time μ_t from p_0^C to p_G^C and standard deviation σ_t for map CORRIDOR M_C

		Actual-path-time-specific metrics	
GP	LP	μ_t [sec]	σ_t [sec]
navfn	dwa	47.53	14.85
globalplanner	dwa	55.98	24.87
sbpl	dwa	78.72	25.80
navfn	**eband**	**107.52**	**0.81**
globalplanner	**eband**	**106.86**	**1.00**
sbpl	eband	70.80	17.93
navfn	**teb**	**44.89**	**0.44**
globalplanner	**teb**	**44.83**	**0.44**
sbpl	**teb**	**46.61**	**0.24**

Combinations of planners that completed all missions and the values of their corresponding metrics are noted with bold

Table 12 Mean actual path length $\mu_l(\mathcal{P})$ and standard deviation $\sigma_l(\mathcal{P})$, and mean $\mu_s(\mathcal{P})$ and standard deviation $\sigma_s(\mathcal{P})$ of the actual path smoothness for map CORRIDOR $\boldsymbol{M}_C$

		Actual-path-specific metrics			
GP	LP	$\mu_l(\mathcal{P})$ [m]	$\sigma_l(\mathcal{P})$ [m]	$\mu_s(\mathcal{P})$ [rad]	$\sigma_s(\mathcal{P})$ [rad]
navfn	dwa	9.24	3.37	1.56	0.16
globalplanner	dwa	8.62	3.23	1.66	0.15
sbpl	dwa	9.12	3.01	1.60	0.23
navfn	**eband**	**20.15**	**0.09**	**2.36**	**0.01**
globalplanner	**eband**	**20.04**	**0.07**	**2.36**	**0.01**
sbpl	eband	12.79	3.52	1.78	0.27
navfn	**teb**	**20.87**	**0.03**	**1.66**	**0.06**
globalplanner	**teb**	**20.88**	**0.04**	**1.69**	**0.09**
sbpl	**teb**	**22.99**	**0.06**	**1.65**	**0.02**

Combinations of planners that completed all missions and the values of their corresponding metrics are noted with bold

simulations for all combination of global and local planners in map $\boldsymbol{M}_C$.

In terms of time taken to reach the goal pose p_G^c from p_0^C (Table 11), all combinations of global planners with dwa_local_planner are excluded from evaluation (since it's a condition that the robot reach its goal), and the same applies to the combination of sbpl_lattice_planner and eband_local_planner. The remaining combinations illustrate (a) that under teb_local_planner the robot takes the least amount of time to traverse the path from p_0 to p_G (which is to be expected, since it approaches the problem of navigation in terms of optimising with respect to time), consequently (b) that eband_local_planner is the slowest among the two − with a significant margin since it takes more than twice as much time to complete a mission −, and (c) that the former's travel times are the most consistent. The fact that sbpl_lattice_planner produces plans of greater length − approximately 17% lengthier than those of the other two global planners (Table 8) − made its combination with teb_local_planner have an impact on the robot's travel time from start to finish, with a magnitude of about over two seconds, which translates to roughly a 10% increase in travel times compared to those of teb_local_planner with navfn and global_planner.

In terms of the traversed paths' mean length (Table 12), the same combinations of global planners with eband_local_planner and teb_local_planner make the robot travel greater lengths compared to their global plans: both approaches deform the global plan in order to gain more clearance from obstacles, and this is the reason why dwa_local_planner fails in every single simulation. Furthermore, the paths that teb_local_planner dictated were the longest but the most consistent, and the most consistent of them all were observed when navfn was used as the global planner, which is to be expected since the plans produced by it are deterministic. In terms of path

Table 13 Overall minimum distance of the actual paths $\mathcal{P}$ the robot took from any obstacle across all experiments $\inf(d(\mathcal{P}, \boldsymbol{M}_C))$, mean minimum distance $\mu(d(\mathcal{P}, \boldsymbol{M}_C))$ and mean standard deviation $\sigma(d(\mathcal{P}, \boldsymbol{M}_C))$ from all obstacles for map CORRIDOR $\boldsymbol{M}_C$

		Actual-path-obstacle-specific metrics		
GP	LP	$\inf(d(\mathcal{P}, \boldsymbol{M}_C))$ [m]	$\mu(d(\mathcal{P}, \boldsymbol{M}_C))$ [m]	$\sigma(d(\mathcal{P}, \boldsymbol{M}_C))$ [m]
navfn	dwa	0.00 (−0.02)	0.17	0.19
globalplanner	dwa	0.00 (−0.02)	0.15	0.18
sbpl	dwa	0.06	0.21	0.16
navfn	**eband**	**0.07**	**0.55**	**0.27**
globalplanner	**eband**	**0.09**	**0.53**	**0.27**
sbpl	eband	0.10	0.40	0.17
navfn	**teb**	**0.18**	**0.64**	**0.19**
globalplanner	**teb**	**0.18**	**0.64**	**0.20**
sbpl	**teb**	**0.18**	**0.49**	**0.16**

Combinations of planners that completed all missions and the values of their corresponding metrics are noted with bold

smoothness, sbpl_lattice_planner's combination with teb_local_planner exhibited the highest and least-varied path smoothness.

In terms of clearance with regard to obstacles in map $\boldsymbol{M}_C$ (Table 13), the combination of local planner eband_local_planner with global planner sbpl_lattice_planner didn't make the robot collide with obstacles, while its clearance was lower than that of teb_local_planner. The latter's parameterisation feature regarding minimum clearance from obstacles (set at 0.10m) clearly played a significant role in the robot's distancing from obstacles: at 0.18m the planner gave the robot the largest minimum clearance from obstacles among all other local planners and across all experiments (this was a further reason why it scored as much with regard to the qualitative metric of parameterisability − Table 2). Furthermore, the same is observed with regard to the mean distance of each robot pose to the closest obstacle in the map of world CORRIDOR, while its variation is the least (compared to combinations that did complete the mission). As an aside, local planner dwa_local_planner failed to avoid obstacles at least once across N simulations on both occasions where the corresponding global planner was ignorant about the robot's kinematics. On the other hand, when sbpl_lattice_planner was employed as the global planner, the robot did not collide with an obstacle even once. Furthermore, its mean distance to the closest obstacle was the highest among the three global planners employed, while its standard deviation was the lowest.

In terms of the traversed paths' deviation from their corresponding global plans (Table 14), local planner dwa_local_planner exhibited the lowest mean pose deviation for its combinations with global planners that do not account for the robot's kinematics, which is to be expected, since, as discussed earlier, it is the controller of highest fidelity to the global plan among the three. However, due to its inability to complete the mission, all of its combinations with global planners are excluded from further evaluation, and the same applies to its combinations with sbpl_lattice_planner and eband_local_planner. From the rest of the combinations, those employing teb_local_planner feature the least mean deviation of each pose with respect to the global plan, and, in particular, its combination with sbpl_lattice_planner exhibited the least total deviation among all other combinations. Furthermore, its mean discrete Frechet distance was consistently lower than that of its rival eband_local_planner.

Appendix C: Details of Evaluation on Map WILLOWGARAGE

1) Global-planner-specific evaluation

Tables 15 and 16 illustrate the resulted values of the quantitative metrics concerning global planners defined in Table 1 over $N = 10$ simulations in environment WILLOWGARAGE.

Regarding the produced plans (Table 15), and with respect to the metrics concerning the evaluation of global planners nothing changed compared to those concerning map $\boldsymbol{M}_C$: global_planner again produced paths of the least length, sbpl_lattice_planner those of the most length and of the lowest density but of the highest smoothness, and navfn produced the least coarse plans, but those of the lowest smoothness.

What did change was the overall minimum distance from obstacles of the plans of global planner sbpl_lattice_planner (Table 16): while in the map of world CORRIDOR it never once planned *through* obstacles, in the map of world WILLOWGARAGE it did − as did

Table 14 Mean deviation $\mu_\delta(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$, mean total deviation $\mu_\Delta(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$, and mean Frechet distance $\mu_\delta^F(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$ between the actual paths $\boldsymbol{\mathcal{P}}$ the robot took and their corresponding global plans $\boldsymbol{\mathcal{G}}$ for map CORRIDOR $\boldsymbol{M}_C$

		Actual-path-deviation-specific metrics		
GP	LP	$\mu_\delta(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$ [m]	$\mu_\Delta(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$ [m]	$\mu_\delta^F(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$ [m]
navfn	dwa	0.04	45.98	5.58
globalplanner	dwa	0.04	39.39	5.28
sbpl	dwa	0.09	101.28	5.08
navfn	**eband**	**0.10**	**104.96**	**0.38**
globalplanner	**eband**	**0.13**	**139.69**	**0.47**
sbpl	eband	0.13	145.06	5.06
navfn	**teb**	**0.07**	**81.96**	**0.26**
globalplanner	**teb**	**0.08**	**89.89**	**0.29**
sbpl	**teb**	**0.07**	**73.59**	**0.26**

Combinations of planners that completed at least one mission and the values of their corresponding metrics are noted with bold

Table 15 Mean global plan length $\mu_l(\mathcal{G})$ and standard deviation $\sigma_l(\mathcal{G})$, mean global plan resolution $\mu_r(\mathcal{G})$, and mean $\mu_s(\mathcal{G})$ and standard deviation $\sigma_s(\mathcal{G})$ of the global plans' smoothness for map WILLOWGARAGE $\boldsymbol{M}_W$

		Global-planner-specific metrics				
GP	LP	$\mu_l(\mathcal{G})$ [m]	$\sigma_l(\mathcal{G})$ [m]	$\mu_r(\mathcal{G})$ [poses/m]	$\mu_s(\mathcal{G})$ [rad]	$\sigma_s(\mathcal{G})$ [rad]
navfn	dwa	44.50	0.02	37.15	1.99	0.00
navfn	eband	44.50	0.02	37.15	1.99	0.00
navfn	teb	44.53	0.04	37.10	1.99	0.00
globalplanner	dwa	44.48	0.00	36.76	1.97	0.00
globalplanner	eband	44.49	0.01	36.61	1.97	0.00
globalplanner	teb	44.49	0.01	36.64	1.97	0.00
sbpl	dwa	48.01	0.00	30.93	2.02	0.00
sbpl	eband	48.01	0.00	30.93	2.02	0.00
sbpl	teb	48.01	0.01	30.95	2.02	0.00

global_planner again. Except for this, it again exhibited the lowest mean minimum distance from obstacles and the most consistency around it. On the other hand, navfn gained, on average, a centimeter of clearance, and its performance with respect to the mean minimum distance of each pose from obstacles was somewhat equivalent to that of global_planner's, as was also its standard deviation.

2) Local-planner-specific evaluation

Table 17 summarises the resulted values of the quantitative metrics concerning local planners defined in Table 1 over N simulations.

The increased level of navigation arduousness of the WILLOWGARAGE world exposed more of the local planners' shortcomings. What is impressive is that all combinations of global planners with local planners dwa_local_planner and eband_local_planner failed to traverse the path from the initial to the goal pose in all conducted simulations.

When global planners that do not account for the robot's kinematics were used, the former aborted all missions, and it aborted most of them (7 out of 10) in the contrary case (− it becomes clear that the use of a global planner mindful of the robot's kinematics is advantageous in the case of an inflexible local planner). In the former case it again exhibited the highest number of control failures and, in the latter, the highest mean number of costmap clearances.

With regard to the performance of eband_local_planner in map $\boldsymbol{M}_W$, the same as in map $\boldsymbol{M}_C$ applies for it in the case of its combination with sbpl_lattice_planner: it exhibited the lowest number of path failures (at least three times lower than the next lower-most combination). Although eband_local_planner managed to make the robot travel significantly larger distances compared to dwa_local_planner (Fig. 8), and although it was consistent in its navigation (the software bug mentioned in the previous subsection concerning its combination with sbpl_lattice_planner did not emerge in map $\boldsymbol{M}_W$),

Table 16 Overall minimum distance of the global plans from any obstacle across all experiments $\inf(d(\mathcal{G}, \boldsymbol{M}_W))$, mean minimum distance $\mu(d(\mathcal{G}, \boldsymbol{M}_W))$ and standard deviation $\sigma(d(\mathcal{G}, \boldsymbol{M}_W))$ from all obstacles for map WILLOWGARAGE $\boldsymbol{M}_W$

		Global-planner-obstacle-specific metrics		
GP	LP	$\inf(d(\mathcal{G}, \boldsymbol{M}_W))$ [m]	$\mu(d(\mathcal{G}, \boldsymbol{M}_W))$ [m]	$\sigma(d(\mathcal{G}, \boldsymbol{M}_W))$ [m]
navfn	dwa	0.01	0.51	0.52
navfn	eband	0.01	0.51	0.52
navfn	teb	0.01	0.51	0.52
globalplanner	dwa	0.00 (−0.02)	0.51	0.53
globalplanner	eband	0.00 (−0.02)	0.51	0.53
globalplanner	teb	0.00 (−0.02)	0.51	0.53
sbpl	dwa	0.00 (−0.02)	0.35	0.43
sbpl	eband	0.00 (−0.02)	0.35	0.43
sbpl	teb	0.00 (−0.02)	0.35	0.43

Table 17 Mean number of aborted missions over the number of simulations conducted μ_A/N, mean number of rotation recoveries μ_{RR} and their standard deviation σ_{RR}, mean number of costmap clearances μ_{CC} and their standard deviation σ_{CC}, mean number of path failures μ_{PF} and their standard deviation σ_{PF}, and mean number of path failures over the mean number of local planner calls μ_{PF}/μ_{LPC} for all combinations of global and local planners featured in Table 3 on map WILLOWGARAGE $\boldsymbol{M}_W$

		Local-planner-specific metrics							
GP	LP	μ_A/N	μ_{RR}	σ_{RR}	μ_{CC}	σ_{CC}	μ_{PF}	σ_{PF}	μ_{PF}/μ_{LPC}
navfn	dwa	1.00	2.00	0.00	3.00	0.00	45.80	10.97	0.09
globalplanner	dwa	1.00	2.30	0.48	3.00	0.00	35.40	10.94	0.08
sbpl	dwa	0.70	3.20	1.48	3.60	1.07	12.00	4.85	0.03
navfn	eband	0.00	0.00	0.00	0.00	0.00	20.10	25.78	N/A
globalplanner	eband	0.00	0.00	0.00	0.00	0.00	8.00	10.87	N/A
sbpl	eband	0.00	0.00	0.00	0.00	0.00	3.20	2.15	N/A
navfn	teb	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
globalplanner	teb	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sbpl	teb	0.00	0.00	0.00	0.00	0.00	0.10	0.32	0.00

it required more than the specified amount of time on every simulation, again exhibiting its overly-safe approach (its mean times were consistently slow, as previously observed in its simulations in map $\boldsymbol{M}_C$). Its mean travel times are illustrated in Table 18 (recount that $t_W^{max} = 180$ sec).

In contrast to all combinations of global planners with `dwa_local_planner` and `eband_local_planner`, all combinations with `teb_local_planner` managed to travel a path from the initial to the goal pose. Again, it championed in not aborting a mission even once, in reaching the goal pose on all simulations, in not executing a single recovery behaviour, and only its combination with global planner `sbpl_lattice_planner` failed to procure valid motor inputs, but only minimally (− the use of a global planner mindful of the robot's kinematics seems not to be as advantageous in the case of a flexible local planner as it is in that of an inflexible one).

Table 18 Mean travel time μ_t from p_0^W to p_G^W and standard deviation σ_t for map WILLOWGARAGE $\boldsymbol{M}_W$

		Actual-path-time-specific metrics	
GP	LP	μ_t [sec]	σ_t [sec]
navfn	dwa	20.46	19.24
globalplanner	dwa	22.76	18.85
sbpl	dwa	78.91	48.12
navfn	eband	158.14	5.25
globalplanner	eband	151.23	2.00
sbpl	eband	147.53	2.46
navfn	**teb**	**95.45**	**0.34**
globalplanner	**teb**	**95.50**	**0.41**
sbpl	**teb**	**100.55**	**1.56**

Combinations of planners that completed all missions and the values of their corresponding metrics are noted with bold

3) Global/local-planners-combination-specific evaluation

Tables 18, 19, 20, and 21 summarise the resulted values of the quantitative metrics concerning the combination of global and local planners defined in Table 1 over N simulations for all combination of global and local planners in the map of world WILLOWGARAGE.

In terms of time taken to reach the goal pose p_G^W from p_0^W (Table 18), all combinations of global planners with `dwa_local_planner` and `eband_local_planner` are excluded from evaluation due to their inability to make the robot travel the whole path and reach p_G^W. − However, it should be noted that while `dwa_local_planner`'s combination with global planners ignorant of the robot's kinematics could not even pass through the first challenging opening on walls (a door), its combination with `sbpl_lattice_planner` managed to do so, and do it for the next four openings, before caving in. The remaining combinations − all of them having `teb_local_planner` as their local planner − make the robot take just over half the allocated maximum time to travel the entirety of the path from p_0^W to p_G^W. The fact that `sbpl_lattice_planner` produces plans of greater length − approximately 4.5% lengthier than those of the other two global planners (Table 15) − made its combination with `teb_local_planner` have an impact on the robot's travel time from start to finish, with a magnitude of just over 5 seconds, which translates to roughly the same (5%) increase in travel times compared to those of `teb_local_planner` with `navfn` and `global_planner`.

Table 19 Mean actual path length $\mu_l(\mathcal{P})$ and standard deviation $\sigma_l(\mathcal{P})$, and mean $\mu_s(\mathcal{P})$ and standard deviation $\sigma_s(\mathcal{P})$ of the actual path smoothness for map WILLOWGARAGE $\boldsymbol{M}_W$

		Actual-path-specific metrics			
GP	LP	$\mu_l(\mathcal{P})$ [m]	$\sigma_l(\mathcal{P})$ [m]	$\mu_s(\mathcal{P})$ [rad]	$\sigma_s(\mathcal{P})$ [rad]
navfn	dwa	2.35	0.03	0.67	0.13
globalplanner	dwa	2.35	0.03	0.63	0.17
sbpl	dwa	7.94	7.62	0.81	0.35
navfn	eband	29.80	1.26	1.83	0.02
globalplanner	eband	28.83	0.43	1.84	0.01
sbpl	eband	26.98	0.53	1.78	0.02
navfn	**teb**	**46.53**	**0.08**	**1.57**	**0.04**
globalplanner	**teb**	**46.55**	**0.04**	**1.61**	**0.02**
sbpl	**teb**	**48.73**	**0.09**	**1.61**	**0.01**

Combinations of planners that completed all missions and the values of their corresponding metrics are noted with bold

In terms of the traversed paths' mean length (Table 19), it is again observed that, due to the deformation of the global plan by `teb_local_planner` so that the set minimum obstacle clearance is achieved, the actual paths are longer than their corresponding global plans; and this to a degree of roughly 4.5% with regard to `navfn` and `global_planner`, and 1.5% with regard to `sbpl_lattice_planner`. In terms of path smoothness, its combination with `navfn` gave the smoothest paths, with the other two combinations following closely behind.

In terms of clearance with regard to obstacles in map $\boldsymbol{M}_W$ (Table 20), `teb_local_planner` just about managed to achieve the minimum set robot-obstacle distance (set as before to 0.10m) when combined with ROS's default planners; when combined with `sbpl_lattice_planner`, however, it failed to do so. Its combination with global planner `navfn` gave the robot the largest mean clearance from obstacles across all simulations; its combination with `sbpl_lattice_planner` gave it the least mean clearance and the least variation around it (5cm less), a behaviour consistent with that exhibited in the map of world CORRIDOR.

In terms of the traversed paths' deviation from their corresponding global plans (Table 21), no consistency is observed compared to the results regarding $\boldsymbol{M}_C$, while `teb_local_planner`'s combination with `global_planner` exhibited the largest mean, total mean deviation and mean discrete Frechet distance in $\boldsymbol{M}_C$, in $\boldsymbol{M}_W$ it exhibited the least; and its combination with `sbpl_lattice_planner` gave the largest mean discrete Frechet distance, when in $\boldsymbol{M}_C$ it gave the least.

D. Details of evaluation on map CSAL

1) Global-planner-specific evaluation

Tables 22 and 23 illustrate the resulted values of the quantitative metrics concerning global planners defined in

Table 20 Overall minimum distance of the actual paths $\mathcal{P}$ the robot took from any obstacle across all experiments $\inf(d(\mathcal{P}, \boldsymbol{M}_W))$, mean minimum distance $\mu(d(\mathcal{P}, \boldsymbol{M}_W))$ and mean standard deviation $\sigma(d(\mathcal{P}, \boldsymbol{M}_W))$ from all obstacles for map WILLOWGARAGE $\boldsymbol{M}_W$

		Actual-path-obstacle-specific metrics		
GP	LP	$\inf(d(\mathcal{P}, \boldsymbol{M}_W))$ [m]	$\mu(d(\mathcal{P}, \boldsymbol{M}_W))$ [m]	$\sigma(d(\mathcal{P}, \boldsymbol{M}_W))$ [m]
navfn	dwa	0.33	0.37	0.14
globalplanner	dwa	0.30	0.38	0.14
sbpl	dwa	0.01	0.36	0.23
navfn	eband	0.03	0.61	0.51
globalplanner	eband	0.01	0.61	0.52
sbpl	eband	0.00	0.65	0.53
navfn	**teb**	**0.10**	**0.82**	**0.47**
globalplanner	**teb**	**0.10**	**0.79**	**0.48**
sbpl	**teb**	**0.09**	**0.67**	**0.42**

Combinations of planners that completed all missions and the values of their corresponding metrics are noted with bold

Table 21 Mean deviation $\mu_\delta(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$, mean total deviation $\mu_\Delta(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$, and mean Frechet distance $\mu_\delta^F(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$ between the actual paths $\boldsymbol{\mathcal{P}}$ the robot took and their corresponding global paths $\boldsymbol{\mathcal{G}}$ for map WILLOWGARAGE $\boldsymbol{M}_W$

		Actual-path-deviation-specific metrics		
GP	LP	$\mu_\delta(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$ [m]	$\mu_\Delta(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$ [m]	$\mu_\delta^F(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{G}})$ [m]
navfn	dwa	0.14	226.98	34.12
globalplanner	dwa	0.12	185.54	34.06
sbpl	dwa	0.27	457.42	31.29
navfn	eband	0.12	194.53	11.98
globalplanner	eband	0.12	190.02	12.74
sbpl	eband	0.18	271.59	15.89
navfn	**teb**	**0.11**	**174.18**	**0.43**
globalplanner	**teb**	**0.10**	**152.02**	**0.43**
sbpl	**teb**	**0.11**	**162.33**	**0.54**

Combinations of planners that completed at least one mission and the values of their corresponding metrics are noted with bold

Table 1 over $N = 10$ real-world experiments in environment CSAL.

Regarding the produced plans (Table 22), and with respect to metrics metrics concerning the evaluation of global planners, `navfn` produced paths of the least length, `sbpl_lattice_planner` those of the most length, of the lowest density, and of the highest variability, but of the highest smoothness; and `global_planner` produced the least coarse plans.

Global planner `sbpl_lattice_planner` exhibited the same behaviour as that in simulated world WILLOGARAGE: the overall minimum distance from obstacles of its plans (Table 23) was zero, and so was that of `global_planner`, which in all three study cases consistently planned through obstacles. Except for this, `sbpl_lattice_planner` exhibited again the lowest mean minimum distance from obstacles and the most consistency around it. On the other hand, `navfn` produced the best plans with respect to mean distance from obstacles, but those of the largest inconsistency among the three global planners.

2) Local-planner-specific evaluation

Table 24 summarises the resulted values of the quantitative metrics concerning local planners defined in Table 1 over N simulations.

What is impressive here is that all combinations of global planners with local planner `dwa_local_planner` again failed to traverse the path from the initial to the goal pose in all conducted experiments.

With regard to the performance of `teb_local_planner` in environment CSAL, it again championed — never aborting any mission, and making no recovery attempts, although it experienced a minimal number of path failures. As for `eband_local_planner`, it exhibited

Table 22 Mean global plan length $\mu_l(\boldsymbol{\mathcal{G}})$ and standard deviation $\sigma_l(\boldsymbol{\mathcal{G}})$, mean global plan resolution $\mu_r(\boldsymbol{\mathcal{G}})$, and mean $\mu_s(\boldsymbol{\mathcal{G}})$ and standard deviation $\sigma_s(\boldsymbol{\mathcal{G}})$ of the global plans' smoothness for map CSAL $\boldsymbol{M}_L$

		Global-planner-specific metrics				
GP	LP	$\mu_l(\boldsymbol{\mathcal{G}})$ [m]	$\sigma_l(\boldsymbol{\mathcal{G}})$ [m]	$\mu_r(\boldsymbol{\mathcal{G}})$ [poses/m]	$\mu_s(\boldsymbol{\mathcal{G}})$ [rad]	$\sigma_s(\boldsymbol{\mathcal{G}})$ [rad]
navfn	dwa	21.87	0.14	199.84	2.33	0.00
navfn	eband	21.78	0.13	199.96	2.33	0.00
navfn	teb	21.87	0.16	199.95	2.33	0.00
globalplanner	dwa	21.90	0.06	200.06	2.33	0.00
globalplanner	eband	21.89	0.10	200.06	2.33	0.00
globalplanner	teb	21.84	0.13	200.07	2.33	0.00
sbpl	dwa	22.07	0.04	131.61	2.31	0.01
sbpl	eband	22.09	0.10	131.66	2.31	0.01
sbpl	teb	22.12	0.39	133.03	2.30	0.04

Table 23 Overall minimum distance of the global plans from any obstacle across all experiments $\inf(d(\boldsymbol{\mathcal{G}}, \boldsymbol{M}_L))$, mean minimum distance $\mu(d(\boldsymbol{\mathcal{G}}, \boldsymbol{M}_L))$ and standard deviation $\sigma(d(\boldsymbol{\mathcal{G}}, \boldsymbol{M}_L))$ from all obstacles for map CSAL $\boldsymbol{M}_L$

		Global-planner-obstacle-specific metrics		
GP	LP	$\inf(d(\boldsymbol{\mathcal{G}}, \boldsymbol{M}_L))$ [m]	$\mu(d(\boldsymbol{\mathcal{G}}, \boldsymbol{M}_L))$ [m]	$\sigma(d(\boldsymbol{\mathcal{G}}, \boldsymbol{M}_L))$ [m]
navfn	dwa	0.01	0.47	0.42
navfn	eband	0.01	0.47	0.42
navfn	teb	0.01	0.47	0.42
globalplanner	dwa	0.00 (−0.02)	0.45	0.40
globalplanner	eband	0.00 (−0.02)	0.45	0.40
globalplanner	teb	0.00 (−0.02)	0.45	0.41
sbpl	dwa	0.00 (−0.02)	0.41	0.37
sbpl	eband	0.00 (−0.02)	0.41	0.37
sbpl	teb	0.00 (−0.02)	0.41	0.37

minimal recovery attempts, but significant path failures when paired with global planners that do not account for the robot's kinematics.

3) Global/local-planners-combination-specific evaluation

Tables 25, 26, 27, and 28 summarise the resulted values of the quantitative metrics concerning the combination of global and local planners defined in Table 1 over N simulations for all combination of global and local planners in environment CSAL.

In terms of time taken to reach the goal pose p_G^L from p_0^L (Table 25), all combinations of global planners with `dwa_local_planner` are excluded from evaluation due to their inability to make the robot travel the whole path and reach p_G^L. Local planner `teb_local_planner` traversed the assigned paths in the less amount of mean time compared to `eband_local_planner` for the same global planner.

In terms of the traversed paths' mean length (Table 26), `teb_local_planner` did not deform its given global plans to the degree it did in simulations, and this is observable as it produced paths with the least mean length. On the other hand, `eband_local_planner`'s dictated paths were the longest, but most consistent in length. The latter produced paths of lower smoothness, compared to the former, and those of the most consistency with regard to smoothness.

In terms of clearance with regard to obstacles in map $\boldsymbol{M}_L$ (Table 27), `teb_local_planner` did not manage to achieve the minimum set robot-obstacle distance (set as before to 0.10m) when combined with any global planner (its mean value was 0.08m), however, under its control, the robot's mean minimum distance to obstacles and its standard deviation around it were lower than those when local planner `eband_local_planner` was employed. The latter was the only local planner that did achieve to exhibit more than the set robot-obstacle threshold distance, and it did so consistently.

Table 24 Mean number of aborted missions over the number of simulations conducted μ_A/N, mean number of rotation recoveries μ_{RR} and their standard deviation σ_{RR}, mean number of costmap clearances μ_{CC} and their standard deviation σ_{CC}, mean number of path failures μ_{PF} and their standard deviation σ_{PF}, and mean number of path failures over the mean number of local planner calls μ_{PF}/μ_{LPC} for all combinations of global and local planners featured in Table 3 on map CSAL $\boldsymbol{M}_L$

		Local-planner-specific metrics							
GP	LP	μ_A/N_s	μ_{RR}	σ_{RR}	μ_{CC}	σ_{CC}	μ_{PF}	σ_{PF}	μ_{PF}/μ_{LPC}
navfn	dwa	1.00	2.20	0.42	3.00	0.00	37.40	17.85	0.08
globalplanner	dwa	1.00	2.60	0.70	3.20	0.63	30.20	23.66	0.06
sbpl	dwa	1.00	2.40	0.70	3.30	0.95	4.10	3.14	0.01
navfn	eband	0.00	0.60	0.97	0.90	1.45	57.00	26.72	N/A
globalplanner	eband	0.00	1.00	0.67	0.40	0.97	65.00	29.84	N/A
sbpl	eband	0.00	1.40	0.52	1.10	1.37	5.80	4.13	N/A
navfn	teb	0.00	0.00	0.00	0.00	0.00	1.10	0.88	0.00
globalplanner	teb	0.00	0.00	0.00	0.00	0.00	1.40	1.17	0.00
sbpl	teb	0.00	0.00	0.00	0.00	0.00	2.70	3.27	0.00

Table 25 Mean travel time μ_t from p_0^W to p_G^L and standard deviation σ_t for map CSAL $\boldsymbol{M}_L$

GP	LP	Actual-path-time-specific metrics	
		μ_t [sec]	σ_t [sec]
navfn	dwa	47.47	15.29
globalplanner	dwa	56.24	15.44
sbpl	dwa	60.30	22.58
navfn	**eband**	**356.25**	**9.88**
globalplanner	**eband**	**354.89**	**10.05**
sbpl	**eband**	**392.16**	**21.29**
navfn	**teb**	**326.70**	**12.88**
globalplanner	**teb**	**330.25**	**13.77**
sbpl	**teb**	**363.16**	**42.35**

Combinations of planners that completed at least one mission and the values of their corresponding metrics are noted with bold

Table 26 Mean actual path length $\mu_l(\boldsymbol{\mathcal{P}})$ and standard deviation $\sigma_l(\boldsymbol{\mathcal{P}})$, and mean $\mu_s(\boldsymbol{\mathcal{P}})$ and standard deviation $\sigma_s(\boldsymbol{\mathcal{P}})$ of the actual path smoothness for map CSAL $\boldsymbol{M}_L$

GP	LP	Actual-path-specific metrics			
		$\mu_l(\boldsymbol{\mathcal{P}})$ [m]	$\sigma_l(\boldsymbol{\mathcal{P}})$ [m]	$\mu_s(\boldsymbol{\mathcal{P}})$ [rad]	$\sigma_s(\boldsymbol{\mathcal{P}})$ [rad]
navfn	dwa	2.97	1.00	0.58	0.42
globalplanner	dwa	2.65	1.50	1.16	0.51
sbpl	dwa	2.99	1.36	0.79	0.54
navfn	**eband**	**22.81**	**0.12**	**2.32**	**0.01**
globalplanner	**eband**	**22.79**	**0.13**	**2.33**	**0.01**
sbpl	**eband**	**22.78**	**0.13**	**2.32**	**0.01**
navfn	**teb**	**22.71**	**0.18**	**2.35**	**0.02**
globalplanner	**teb**	**22.73**	**0.28**	**2.34**	**0.02**
sbpl	**teb**	**23.47**	**0.87**	**2.30**	**0.04**

Combinations of planners that completed at least one mission and the values of their corresponding metrics are noted with bold

Table 27 Overall minimum distance of the actual paths $\boldsymbol{\mathcal{P}}$ the robot took from any obstacle across all experiments $\inf(d(\boldsymbol{\mathcal{P}}, \boldsymbol{M}_L))$, mean minimum distance $\mu(d(\boldsymbol{\mathcal{P}}, \boldsymbol{M}_L))$ and mean standard deviation $\sigma(d(\boldsymbol{\mathcal{P}}, \boldsymbol{M}_L))$ from all obstacles for map CSAL $\boldsymbol{M}_L$

GP	LP	Actual-path-obstacle-specific metrics		
		$\inf(d(\boldsymbol{\mathcal{P}}, \boldsymbol{M}_L))$ [m]	$\mu(d(\boldsymbol{\mathcal{P}}, \boldsymbol{M}_L))$ [m]	$\sigma(d(\boldsymbol{\mathcal{P}}, \boldsymbol{M}_L))$ [m]
navfn	dwa	0.02	0.24	0.09
globalplanner	dwa	0.02	0.25	0.07
sbpl	dwa	0.03	0.26	0.04
navfn	**eband**	**0.11**	**0.52**	**0.20**
globalplanner	**eband**	**0.11**	**0.54**	**0.20**
sbpl	**eband**	**0.13**	**0.57**	**0.19**
navfn	**teb**	**0.08**	**0.51**	**0.18**
globalplanner	**teb**	**0.08**	**0.52**	**0.19**
sbpl	**teb**	**0.08**	**0.56**	**0.17**

Combinations of planners that completed all missions and the values of their corresponding metrics are noted with bold

Table 28 Mean deviation $\mu_\delta(\mathcal{P}, \mathcal{G})$, mean total deviation $\mu_\Delta(\mathcal{P}, \mathcal{G})$, and mean Frechet distance $\mu_\delta^F(\mathcal{P}, \mathcal{G})$ between the actual paths P the robot took and their corresponding global plans G for map CSAL $\boldsymbol{M}_L$

		Actual-path-deviation-specific metrics		
GP	LP	$\mu_\delta(\mathcal{P}, \mathcal{G})$ [m]	$\mu_\Delta(\mathcal{P}, \mathcal{G})$ [m]	$\mu_\delta^F(\mathcal{P}, \mathcal{G})$ [m]
navfn	dwa	0.03	1.74	12.69
globalplanner	dwa	0.05	3.41	12.43
sbpl	dwa	0.04	2.32	12.58
navfn	**eband**	**0.12**	**48.34**	**0.35**
globalplanner	**eband**	**0.13**	**51.63**	**0.35**
sbpl	**eband**	**0.15**	**62.71**	**0.43**
navfn	**teb**	**0.10**	**40.19**	**0.31**
globalplanner	**teb**	**0.11**	**42.84**	**0.33**
sbpl	**teb**	**0.12**	**50.56**	**0.35**

Combinations of planners that completed at least one mission and the values of their corresponding metrics are noted with bold

In terms of the traversed paths' deviation from their corresponding global plans (Table 28), `teb_local_planner` produced paths of the lowest mean and total deviation, which is consistent with the mean minimum robot-obstacle distance values exhibited. Conversely, `eband_local_planner`, consistent with its own mean minimum robot-obstacle distance, produced paths of the greatest mean and total deviation from global plans, and, overall, those of the largest Frechet distance.

References

1. Raghu, D.: RFID 2018–2020: RAIN and NFC Market Status, Outlook and Innovations, IDTEchEx, Presentation, link: https://rainrfid.org/wp-content/uploads/2018/03/IDTechEx-RFID-March-_Distribute.pdf, accessed 11/02/2019
2. Travis, D., Matthew, S., Reynolds, C., Kemp, C.: Finding and navigating to household objects with UHF RFID tags by optimizing RF signal strength, 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, USA. https://doi.org/10.1109/IROS.2014.6942914 (2014)
3. Emilio, D., Francesco, M.: Mobile Robot Localization Using the Phase of Passive UHF RFID Signals. IEEE Trans. Ind. Electron. **61**(1), 365–376 (2014). https://doi.org/10.1109/TIE.2013.2248333
4. Tory-robot, Metralabs, link: https://www.metralabs.com/en/rfid-robot-tory/, accessed 11/02/2019
5. AdvanRobot, Keonn Technologies, link: https://www.keonn.com/systems/view-all-2/inventory-robots.html, accessed 11/02/2019
6. Tally, Simbe Robotics, link: https://www.simberobotics.com/news/simbe-robotics-reveals-rfid-and-machine-learning-capabilities-in-the-latest-iteration-of-their-autonomous-inventory-robot-tally, accessed 11/02/2019
7. Tzitzis, A., Megalou, S., Siachalou, S., Yioultsis, T., Kehagias, A., Tsardoulias, E., Filotheou, A., Symeonidis, A., Petrou, L., Dimitriou, A.G.: Phase ReLock - Localization of RFID Tags by a Moving Robot, to Appear in the European Conference on Antennas and Propagation, Krakow, Poland (2019)
8. Megalou, S., Tzitzis, A., Siachalou, S., Yioultsis, T., Sahalos, J., Tsardoulias, E., Filotheou, A., Symeonidis, A., Petrou, L., Bletsas, A., Dimitriou, A.G.: Fingerprinting Localization of RFID Tags with Real-Time Performance-Assessment, Using a Moving Robot, to Appear in the European Conference on Antennas and Propagation, Krakow, Poland (2019)
9. Gandhi, D., Pinto, L., Gupta, A.: Learning to fly by crashing, 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, pp. 3948–3955. https://doi.org/10.1109/IROS.2017.8206247 (2017)
10. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: An open-source Robot Operating System. ICRA Work. Open Sourc. Softw. **3**(3.2), 5 (2009)
11. Lozano-Prez, T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Commun. ACM **22**(10), 560–570 (1979)
12. Ghosh, S.K.: Visibility algorithms in the plane, Cambridge University Press. ISBN:9780521875745 (2007)
13. Ghosh, S.K., Goswami, P.P.: Unsolved problems in visibility graphs of points, segments and polygons. In: Proceedings of India-Taiwan Conference on Discrete Mathematics, Taipei, pp. 44–54 (2009)
14. Kim, J., Kim, M., Kim, D.: Variants of the quantized visibility graph for efficient path planning. Adv. Robot. **25**(18), 2341–2360 (2011)
15. Bhattacharya, P., Gavrilova, M.L.: Voronoi diagram in optimal path planning. In: 4th International Symposium on Voronoi Diagrams in Science and Engineering, ISVD'07, pp. 38–47 (2007)
16. Garrido, S., Moreno, L., Abderrahim, M., Martin, F.: Path Planning for Mobile Robot Navigation Using Voronoi Diagram and Fast Marching. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2376–2381 (2006)
17. Ok, K.-C., Ansari, S., Gallagher, B., Sica, W., Dellaert, F., Stilman, M.: Path Planning with Uncertainty: Voronoi Uncertainty Fields. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 4596–4601. IEEE (2013)
18. Kavraki, L.E., Svestka, P., Latombe, J.-C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
19. Laumond, J.-P., Nissoux, C.: Visibility-based probabilistic roadmaps for motion planning. In: Proceedings of the IEEE/RSJ

International Conference on Intelligent Robots and Systems, IROS'99, vol. 3, pp. 1316–1321 (1999)
20. Bohlin, R., Kavraki, L.E.: Path planning using lazy PRM. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'00, vol. 1, pp. 521–528 (2000)
21. Hsu, D., Sanchez-Ante, G., Sun, Z.: Hybrid PRM Sampling with a Cost-Sensitive Adaptive Strategy. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation 2005, ICRA 2005, pp. 3874–3880
22. LaValle, S.M.: Rapidly-Exploring Random Trees: A New Tool for Path Planning. TR 98–11, Computer Science Department, Iowa State University (1998)
23. Bruce, J., Veloso, M.: Real-Time Randomized Path Planning for Robot Navigation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2383–2388 (2002)
24. Martin, S.R., Wright, S.E., Sheppard, J.W.: Offline and online evolutionary Bi-Directional RRT algorithms for efficient Re-Planning in dynamic environments (2007)
25. Karaman, S., Frazzoli, E.: Incremental Sampling-based Algorithms for Optimal Motion Planning coRR (2010)
26. Guitton, J., Farges, J.-L., Chatila, R.: Cell-RRT: Decomposing the environment for better plan. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009, pp. 5776–5781 (2009)
27. Jaillet, L., Corts, J., Simon, T.: Sampling-based path planning on configuration-space costmaps. IEEE Trans. Robot. **26**(4), 635–646 (2010)
28. Pivtoraiko, M., Kelly, A.: Efficient constrained path planning via search in state lattices, The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space, 2005, keywords: constrained path planning, planetary rover, rough terrain
29. Latombe, J.-C.: Robot Motion Planning (1991)
30. Koren, Y., Borenstein, J.: Potential field methods and their inherent limitations for mobile robot navigation. In: Proceedings. 1991 IEEE International Conference on Robotics and Automation, vol. 2, pp. 1398–1404, Sacramento, CA. https://doi.org/10.1109/ROBOT.1991.131810 (1991)
31. Ge, S.S., Cui, Y.J.: New potential functions for mobile robot path planning. IEEE Trans. Robot. Autom. **16**(5), 615–620 (2000). https://doi.org/10.1109/70.880813
32. Kowalczyk, W.: Rapid Navigation Function Control for Two-Wheeled Mobile Robots, Journal of Intelligent & Robotic Systems. https://doi.org/10.1007/s10846-018-0879-4 (2018)
33. Park, J.J., Grizzle, J.W.: Graceful Navigation for Mobile Robots in Dynamic and Uncertain Environments (2016)
34. Borenstein, J., Koren, Y.: The vector field histogram-fast obstacle avoidance for mobile robots. IEEE Trans. Robot. Autom. **7**(3), 278–288 (1991)
35. Ulrich, I., Borenstein, J.: VFH+: Reliable obstacle avoidance for fast mobile robots. In: 1998. Proceedings. 1998 IEEE International Conference on Robotics and Automation, vol. 2, pp. 1572–1577. IEEE (1998)
36. Ulrich, I., Borenstein, J.: VFH$^{\hat{*}}$: Local Obstacle avoidance with look-ahead verification. In: ICRA, pp. 2505–2511 (2000)
37. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robot. Autom. Mag. **4**(1), 23–33 (1997)
38. Tovar, B., Guilamo, L., LaValle, S.M.: Gap navigation trees: Minimal representation for visibility-based tasks. In: Algorithmic Foundations of Robotics VI, pp. 425–440. Springer, Berlin (2004)
39. Minguez, J., Montano, L.: Nearness diagram navigation (nd): A new real time collision avoidance approach. In: 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2094–2100. IEEE (2000)
40. Minguez, J., Osuna, J., Montano, L.: A divide and conquer" strategy based on situations to achieve reactive collision avoidance in troublesome scenarios. In: 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on Robotics and Automation, vol. 4, pp. 3855–3862. IEEE (2004)
41. Quinlan, S., Khatib, O.: Elastic bands: Connecting path planning and control. In: 1993. Proceedings., 1993 IEEE International Conference on Robotics and Automation, pp. 802–807. IEEE (1993)
42. Kurniawati, H., Fraichard, T.: From path to trajectory deformation," 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 159–164, San Diego, CA. https://doi.org/10.1109/IROS.2007.4399235 (2007)
43. Rösmann, C., Feiten, W., Wösch, T., Hoffmann, F., Bertram, T.: Trajectory modification considering dynamic constraints of autonomous robots. In: 7th German Conference on Robotics; Proceedings of ROBOTIK 2012, pp. 1–6. VDE (2012)
44. Rösmann, C., Hoffmann, F., Bertram, T.: Kinodynamic trajectory optimization and control for car-like robots. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5681–5686, Vancouver, BC. https://doi.org/10.1109/IROS.2017.8206458 (2017)
45. Fox, D.: KLD-Sampling: Adaptive Particle Filters and Mobile Robot Localization. In: Advances in Neural Information Processing Systems (NIPS) (2001)
46. Philippsen, R.: Motion planning and obstacle avoidance for mobile robots in highly cluttered dynamic environments, EPFL, http://infoscience.epfl.ch/record/33615, https://doi.org/10.5075/epfl-thesis-3146 (2004)
47. Chitta, S., Sucan, I., Cousins, S.: Moveit![ROS topics]. IEEE Robot. Autom. Mag. **19**(1), 18–19 (2012)
48. Likhachev, M., Gordon, G., Thrun, S.: ARA*: Anytime A*with Provable Bounds On Sub-Optimality. Proc. Adv. Neural Inf. Process. Syst. **16**(NIPS), 258–265 (2003)
49. Likhachev, M., Ferguson, D.I., Gordon, G.J., Stentz, A., Thrun, S.: Anytime dynamic a*: an anytime, replanning algorithm. In: ICAPS, pp. 262–271 (2005)
50. Phillips, M., Likhachev, M.: SIPP: Safe interval path planning for dynamic environments. 2011 IEEE International Conference on Robotics and Automation, pp. 5628–5635 (2011)
51. Phillips, M., Likhachev, M.: Sipp: Safe interval path planning for dynamic environments. In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 5628–5635. IEEE (2011)
52. Fréchet, M.M.: Sur quelques points du calcul fonctionnel. Rend. Circ. Mat. Palermo (1884-1940) **22**(1), 1–72 (1906). issn: 0009-725X. https://doi.org/10.1007/BF03018603
53. Tyrrell Rockafellar, R., Wets, R.J.-B.: Variational Analysis. Springer, Berlin. 978-3-540-62772-2 (1998)

Alexandros Filotheou holds a diploma in Electrical and Computer Engineering from the Aristotle University of Thessaloniki, Greece (2013), and Master's degree in Systems, Control, and Robotics from the Royal Institute of Technology (KTH), Stockholm, Sweden (2017). He was a member of the robotics team PANDORA (2014), which, in the past, competed in the international competition RoboCup-RoboRescue, where it attained second place in China and Netherlands. His work experience comprises the design and implementation of applications relevant to Data Bases (Egnatia Odos S.A, 2011). His research includes publications in the International Journal of Control and the European Control Conference. His current professional activity focuses on designing, implementing, and deploying robotics applications, with emphasis on localisation algorithms.

Dr. Emmanouil Tsardoulias has obtained his doctorate and engineering diploma from the Department of Electrical and Computer Engineering at Aristotle University of Thessaloniki (AUTH), Greece, in 2013 and 2007 respectively. His working experience includes participation in the "Eudoxus" project with the collaboration of Ministry of Education and the EU project RAPP (FP7-ICT-610947) among others. His research interests are focused in Robotics and specifically in Autonomous Robots (ground or aerial). Some of the topics involved are autonomous navigation, SLAM (Simultaneous Localization and Mapping), multi-robot exploration / full coverage, robotic architectures oriented for the creation of robotic applications and robot-agnostic RESTful APIs. In addition, from 2009 till now, is the technical manager of the artificial intelligence group of the P.A.N.D.O.R.A. and R4A robotics teams, which operate at the School of Electrical and Computer Engineering, Aristotle University of Thessaloniki. PANDORA has participated in five world-wide RoboCup competitions, achieving the second place in the Autonomy finals twice.

Dr. Antonis Dimitriou received the diploma and the Ph.D degree from the School of Electrical and Computer Engineering of the Aristotle University of Thessaloniki, Greece, in 2001, and 2006 respectively. He is currently with the School of Electrical and Computer Engineering of AUTh, as a teaching and research faculty member. He has participated in more than 20 research projects, 8 of which since 2007 as a principal investigator in the fields of Robotics, RFIDs, and Wireless Sensor Networks. He is currently the coordinator of the project "RELIEF" (http://relief.web.auth.gr/), where prototype SLAM-capable terrestrial robots and drones are designed and constructed to continuously perform inventorying and localization of RFID-tagged items. He was a Management Committee Member in the ICT COST Action IC301 "Wireless Power Transmission for Sustainable Electronics (WiPE)". He is the author or co-author of approximately 60 journal and conference papers. Dr. Dimitriou was the recipient of the Ericsson Award of Excellence in Telecommunications in 2001 and co-recipient of the student-paper award in the 2011 IEEE RFID-TA conference. He received the "IEEE Wireless Communications Letters Exemplary Reviewer" award in 2012 and 2014. He is a Senior IEEE Member since February 2014. He also serves as a reviewer for major journals and as a TPC member for international conferences.

Dr. Andreas Symeonidis is an Associate Professor with the Department of Electrical and Computer Engineering at the Aristotle University of Thessaloniki, Greece and the Chief Research Officer at Cyclopt.com. His research interests include Software engineering processes, Model-driven engineering, Software quality and Software analytics, Middleware Robotics and Knowledge extraction from big data repositories. Dr. Symeonidis' work has been published in over 100 papers, book chapters, and conference publications. He is co-author of the books "Agent Intelligence through Data Mining" (Springer publishing) and "Practical Machine Learning in R" (Leanpub publishing). He has been Project Coordinator of project S-CASE (FP7-ICT-610717) and RAPP (FP7-ICT-610947) and Technical lead in the H2020 project SEAF (H2020-696023). He is currently coordinating more than 10 contract R&D projects. More at: http://users.auth.gr/symeonid.

Dr. Loukas Petrou Associate Professor at the Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki Greece, received the Diploma in Electrical Eng., from the University of Patras Greece, the M. Sc. in Theory and Practice of Automatic Control from U.M.I.S.T., Manchester U.K. and the Ph.D. in Elec. Eng., from Imperial College, London U.K. His research interests include microprocessor applications, autonomous mobile robots, intelligent control of waste water treatment plants, reconfigurable embedded systems, evolutionary computation, fuzzy logic and neural network control systems.