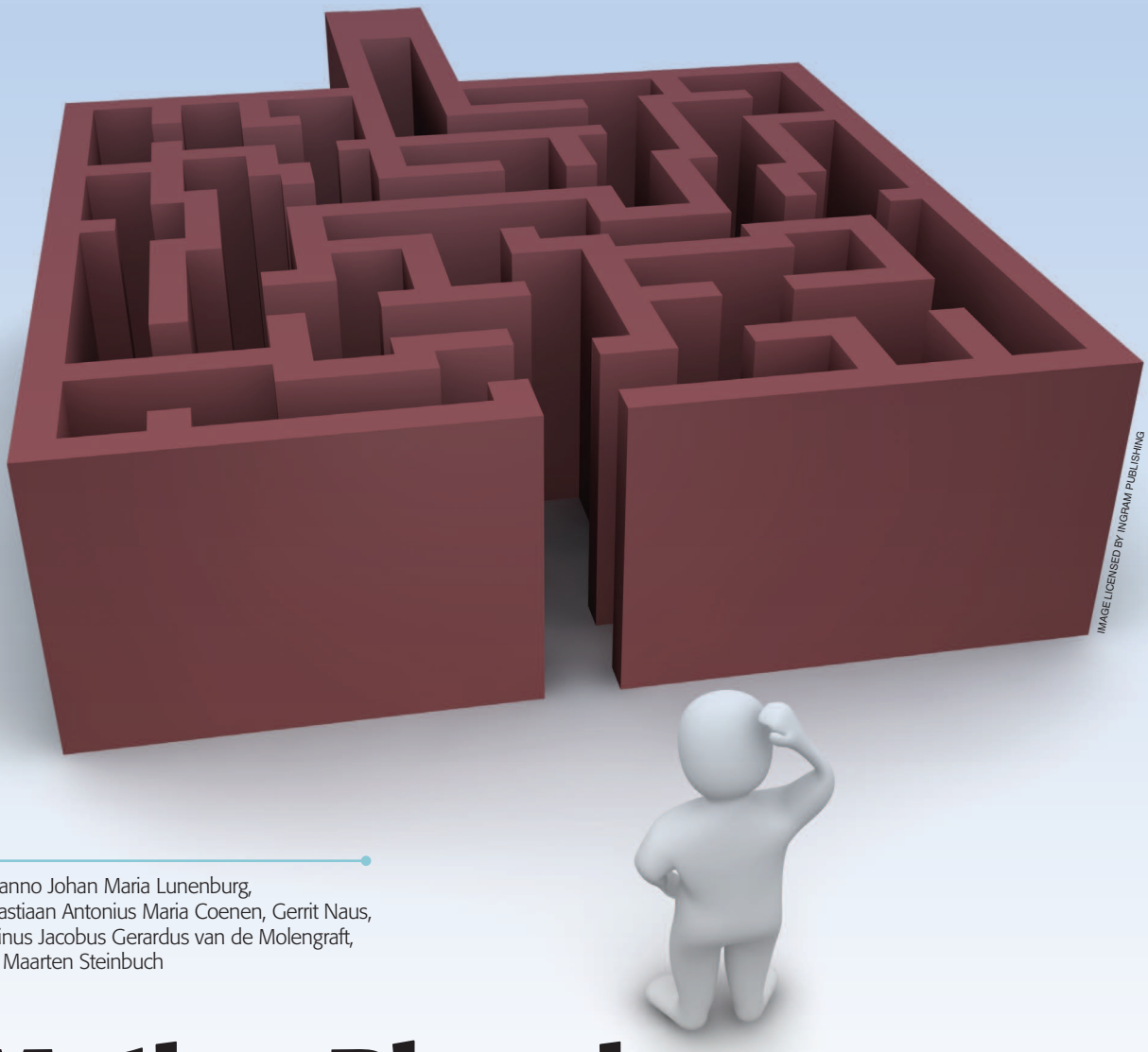

By Janno Johan Maria Lunenburg, Sebastiaan Antonius Maria Coenen, Gerrit Naus, Marinus Jacobus Gerardus van de Molengraft, and Maarten Steinbuch

Motion Planning for Mobile Robots

A Method for the Selection of a Combination of Motion-Planning Algorithms

A motion planner for mobile robots is commonly built out of a number of algorithms that solve the two steps of motion planning: 1) representing the robot and its environment and 2) searching a path through the represented environment. However, the available literature on motion planning lacks a generic methodology to arrive at a combination of representations and search algorithm classes for a practical application. This article presents a method to select appropriate algorithm classes that solve both the steps of motion planning and to select a suitable approach to combine those algorithm classes. The method is verified by comparing its outcome with three different motion planners that have been successfully applied on robots in practice.

Digital Object Identifier 10.1109/MRA.2015.2510798
Date of publication: 16 June 2016

Motion Planning

Motion planning considers the planning of a path for a robot in an environment that can contain dynamic and static obstacles and is generally characterized by uncertainty. A motion planner can serve three different goals: 1) mapping, 2) covering, and 3) navigation. The focus of this article is on motion planning for mobile robots moving on the ground plane with navigation as a goal. Motion planning typically consists of representing the

robot and its environment and searching a path in the represented environment [1].

These representations and search algorithms are abundant in [1]–[4]. Furthermore, as a single algorithm generally does not suffice for a practical application, planners consist of combinations of algorithms to satisfy the designer's requirements [2], [5], [6]. Because the combinations of possibly approximate algorithms are deemed necessary in practice, various approaches have been presented in [2], [4], and [6].

Such motion planners give insight into the choice for a combination of algorithms with respect to predefined requirements.

However, the available literature lacks a generic methodology to arrive at a combination of representations and search algorithms that can handle all the requirements. The selection of an appropriate combination of representations and search algorithms depends on the desired behavior as specified by the designer, the dynamics and the kinematics of the robot at hand, the dynamics of the environment at hand, and the presence of uncertainty. Typically, these conditions are translated into seven requirements, serving as a basis for the selection of these algorithms.

- *Completeness*: The guarantee that either a solution is returned if one exists or failure is returned if no solution exists. Completeness commonly refers only to search algorithms but, in fact, also concerns the representations because poor design choice for the representation may prohibit any search algorithm from returning a solution even though one exists.
- *Optimality*: A solution can be optimal according to criteria such as distance, time, energy use, and so on.
- *Correctness*: The guarantee that if a solution is returned, it will lead the robot to its goal.
- *Dealing with kinodynamic constraints*: A robot has dynamic constraints, e.g., a limited acceleration, and can also be kinematically constrained, e.g., a car cannot move sideways.
- *Robustness against a dynamic environment*: The environment changes if obstacles can move.
- *Robustness against uncertainty*: The representation of the robot and obstacles is generally uncertain, and the environment can be (partially) unknown.
- *Computational complexity*: A measure of memory usage and time to acquire a solution. Keeping the computational complexity tractable usually comes at the expense of one or more of the previously mentioned criteria.

In practice, a motion planner should be complete and return an optimal and correct solution in real time. However, no motion planner is currently able to achieve all these requirements while satisfying the robot constraints in the presence of moving obstacles and uncertainty [7]. Therefore, the motion-planning problem at hand is often approximated by a simplified and realistic version [2], which introduces a tradeoff between the various requirements.

The small amount of literature that discusses the combinations of representations and search algorithms is not generic. Either a single step of the motion-planning problem, i.e., either the representation or the search algorithm, is discussed, or, in more cases, not all of the requirements are discussed, as some are considered irrelevant for a specific robot or goal. In [8], some combinations of algorithms are discussed that, given a specific representation of the environment, find a path. The work in [9] gives an overview of the algorithms that represent the environment and the algorithms that find a path and discusses combinations of both. How the combined algorithms can deal with robot constraints is not discussed. There is also an overview of combinations of algorithms that find a path for robots with certain kinematic constraints [10]. However, it does not consider how to deal with moving obstacles or a partially known environment. In addition, motion planners that combine multiple algorithms with a focus on unmanned aerial vehicles (UAVs) [6] instead of mobile robots moving only on the ground plane are compared. Moreover, this does not consider how the combined algorithms provide robustness against a dynamic environment. Next, there is an overview that discusses the planners that have covering as a specific goal [11]. These planners assume a mostly static environment and thereby exclude robustness against a dynamic environment. In addition, robot constraints are not considered. There are examples of combinations of multiple algorithms that make a well-performing motion planner in practice for a specific mobile robot [12], [13]. Such motion planners give insight into the choice for a combination of algorithms with respect to predefined requirements, but they do not give generic directions for possible combinations.

This article contributes by presenting a method for the selection of a combination of motion-planning algorithms. This method consists of three parts. In the "Selecting an Approach to Combine Algorithm Classes" section, an appropriate approach to combine representations and search algorithm classes is selected. The "Selecting a Representation Class" section discusses the first of the two steps, while the "Selecting a Search Algorithm Class" section discusses the second step. Our method is intended for either novices in robotics or for experts on one of the many other topics in robotics who require navigation on their mobile robot for a practical application. Carefully considering this method and the design criteria can be a first step in designing a navigation system for a mobile robot.

Selecting an Approach to Combine Algorithm Classes

The available approaches for combining motion-planning algorithms are 1) a coupled approach, 2) a decoupled approach, 3) a hierarchical approach, and 4) a reactive approach. Selecting an appropriate approach to combine motion-planning algorithms is of paramount importance as this system design

serves as a guideline for the subsequent design choices in the "Selecting a Representation Class" and "Selecting a Search Algorithm Class" sections. Many of the design decisions made in this stage use the available environment models, e.g., the availability and uncertainty in these models. The presence and predictability of moving obstacles are discussed in this section. Depending on the requirements of the application, these approaches have to be combined with additional features, such as replanning or a bounded uncertainty region. A method shown in Figure 1, is presented to select an approach based on the requirements that are introduced in the "Motion Planning" section. This section and the "Approaches to Combining Algorithm Classes" and "Method to Select an Approach to Combine Algorithm Classes" sections primarily adopt the terminology used in [1] and [2].

Approaches to Combining Algorithm Classes

First, in trajectory planning or a coupled approach, a robot's kinodynamic constraints are directly satisfied in the search for a solution [14]. These methods search for a solution in the state space $\mathcal{S}$ [14]. A state lattice is an example of a representation for planning in the state space satisfying the robot's kinodynamic constraints [4]. Methods that search directly in the state-time space (ST) are called state × time search methods [15]. The notion of time allows the planner to plan the motions that are optimal in the sense of time in the presence of moving obstacles. A coupled approach is generally computationally expensive [7]. To keep computations tractable, such an approach requires algorithms that exploit some form of approximation to arrive at a solution. The completeness of these approximation algorithms is difficult and impractical to prove [6].

A second approach is to decouple the search for a solution by first searching for a partial solution in the configuration space $\mathcal{C}$. This decoupled approach, also called "path planning," generally consists of the following steps [4]:

- *Path planning*: Obtain a collision-free path in $\mathcal{C}$.
- *Path transformation*: Ensure that a path satisfies any kinematic constraints in $\mathcal{S}$ [10], [14].
- *Trajectory planning*: Use a timing function to transform a path into a trajectory that satisfies dynamic constraints.

If this sequence is required to deal with moving obstacles, decoupling into a path planning part and a motion timing part is also possible [5]. This approach follows the first two steps of the decoupled trajectory planning approach, but the trajectory that is formed in the third step also accounts for moving obstacles. Recent approaches to create trajectories from a path are Covariant Hamiltonian Optimization for Motion Planning and Stochastic Trajectory Optimization for Motion Planning [16], which use optimization techniques to minimize cost function while satisfying the kinodynamic constraints of the robot. Planning in the configuration space $\mathcal{C}$ is computationally less expensive than in the state space $\mathcal{S}$ due to the lower dimension. The inherent consequence is a negative influence on the completeness and optimality of the approach [2]. For example, a path obtained in $\mathcal{C}$ with sharp turns might be infeasible for a nonholonomically constrained car. In addition, a single path, obtained by a search in $\mathcal{C}$ can be transformed into a time-optimal trajectory, but this trajectory is not guaranteed to be globally time optimal.

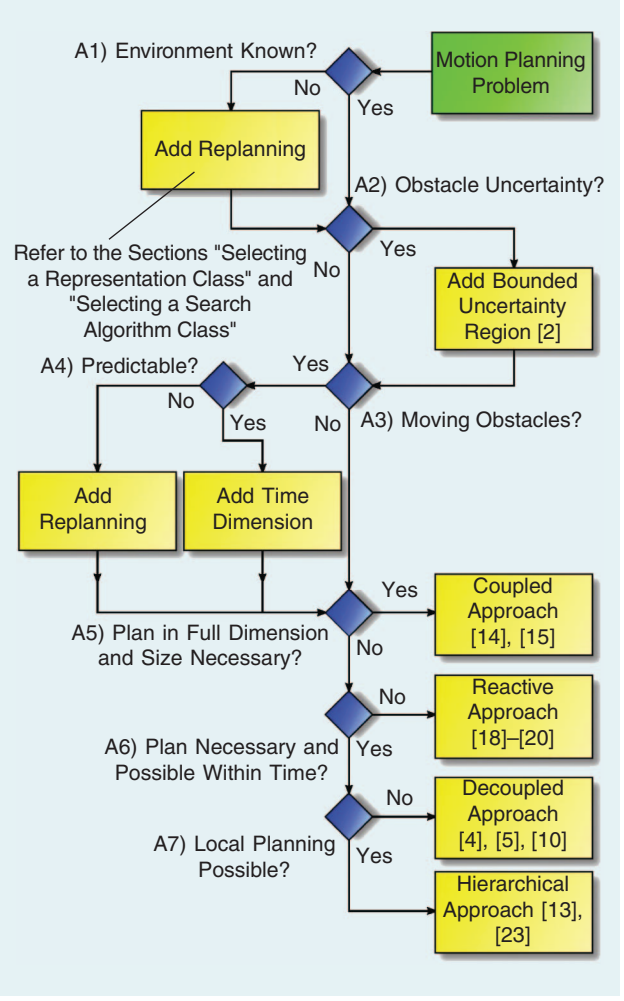

Figure 1. A method resulting in an appropriate approach to combine motion-planning algorithms. The numbers A1–A7 correspond to the questions posed in the "Method to Select an Approach to Combine Algorithm Classes" section.

Third, a hierarchical approach acquires a solution in two or more consecutive methods [6], [9]. This approach uses a planner to find a solution to the goal, using the configuration space or the full-dimensional solution space, followed by a planner that searches for a solution in the vicinity of the robot using the full-dimensional solution space. This is also known as *global motion or path planning* and *local motion planning*. Similar to the decoupled approach, this is a decoupling that sacrifices optimality, but it is not necessarily incomplete [17]. A hierarchical approach is also possible in $\mathcal{C}$. One could first plan a global path that considers only the position and, subsequently, use a local planner that also considers the orientation and, hence, plans in a higher dimension.

The fourth approach is a reactive approach or sensor-based approach. This approach is based on a class of algorithms that are also known as *obstacle avoidance algorithms*.

Rather than searching for a path or trajectory on a certain representation, these compute an instantaneous motion that avoids collision based on the latest sensor information. This approach does not need a representation in the form of a metric map as opposed to the previously discussed approaches. They can be divided into four classes: 1) potential field [18], 2) the first order or velocity obstacle methods [19], 3) the receding horizon control [20], and 4) the (global) nearness diagram [21]. Typically, these methods only compute a desired direction of motion, and hence the amount of planning is reduced to a minimum. However, as such an approach does not consider the planning problem in the entire world space, there is no guarantee that the resulting trajectory will lead to the goal. These local minima are a well-known drawback of the reactive methods. Furthermore, the resulting trajectory is hardly ever optimal. Therefore, a reactive planner is typically used in combination with a global planner, and hence it can be considered as a special form of a hierarchical approach. Besides the selection of a global planner and a reactive planner, the integration of these modules is important. One should consider how the system should react if either of the planners cannot compute a solution, whether a global plan is computing every sample, and so on. Examples of approaches combining global planners with reactive methods can be found in [13].

A path obtained in C with sharp turns might be infeasible for a nonholonomically constrained car.

Method to Select an Approach to Combine Algorithm Classes

The following design criteria, which are numbered A1–A7, form a method to select a suitable approach to combine the motion-planning algorithms. The criteria are based on the requirements that are introduced in the "Motion Planning" section. The method is shown in Figure 1.

A1) *Is the environment known?* In an environment that is not completely known, a plan can become infeasible as sensor data are acquired during execution, and thus a new plan is required. This requires replanning. A replan can be acquired by recomputing the entire representation of the solution space, discussed in the "Method to Select a Representation Class" section, or by updating the representation and performing a new search, as discussed in the "Method to Select a Search Algorithm Class" section.

A2) *Is there uncertainty in the obstacle representation?* Uncertainty in the representation of obstacles, e.g., uncertainty due to imperfect sensors, can be assumed to be contained in a bounded uncertainty region [2]. In practice, a motion planner gains robustness by inflating the obstacle representation with a distance that is equal to the present position uncertainty of obstacles. This robustness, however, comes at the expense of completeness: if obstacles are inflated too much, the planner might not be able to find a solution anymore.

A3) *Does the environment contain moving obstacles?* To be complete and to get an optimal solution in the presence of moving obstacles, it is necessary to represent their trajectories with a time dimension in the solution space. Nevertheless, this significantly increases the computational complexity. Replanning is also possible to deal with a changing environment, but this will result in suboptimal plans and the planner might not be complete. A further alternative to replanning is to update or modify the existing plan. An example of this approach is elastic bands [22].

A4) *Can the motion of obstacles be predicted?* If obstacles are predictable, i.e., have a known trajectory, it is necessary to add a time dimension to the solution space, yielding an ST, to be complete and to ensure a time-optimal solution. A trajectory can be modeled using a prediction model if obstacles have no exactly known trajectory. For example, an obstacle position can be extrapolated in time using a maximum acceleration model. In general, one can say that the effectiveness of including a prediction model depends on the amount of uncertainty in prediction. As the inflation of the obstacle representation increases, the number of possible solutions decreases and the remaining ones will be further from optimality. A time dimension will unnecessarily increase the dimension of the solution space if obstacles are not predictable. Replanning and using a reactive approach are the alternatives to deal with moving obstacles. The inherent consequence is that the solution will not be optimal and the planner is possibly incomplete.

A5) *Is a solution in the form of a plan necessary in the full dimension and size of the solution space?* A coupled approach searches for a solution in the full dimension of the solution space, and, therefore, it can be complete and return optimal solutions. However, the complexity of representation is exponential with the number of dimensions of the solution space. A coupled approach can thus be computationally too complex to achieve a solution within the designer's desired time constraint. The complexity can be negotiated using a hierarchical, reactive, or decoupled approach, but this is generally at the cost of completeness and optimality.

A6) *Is a solution in the form of a plan necessary and possible within a certain time constraint?* Achieving a solution in the form of a plan that avoids collision can be too time-consuming. An environment can change so rapidly due to moving obstacles and uncertainty that a plan can become infeasible before the robot is able to even execute it. In such a case, planning becomes less relevant, and a reactive approach is necessary. As a reactive approach typically only considers the latest sensor information; it is not complete, and its solution is not optimal. One must also question whether an explicit

plan is actually necessary. It can be sufficient to apply a reactive approach that uses the structure of the environment to control the robot to its desired goal pose instead of precomputing the entire motion.

A7) *Local planning possible?* A hierarchical approach is generally preferred, as a local planner in this approach still obtains a solution in the full solution space. Therefore, there is a better chance of finding a solution, and this solution is generally closer to optimality than in the case of a decoupled approach. Despite a lower computational complexity, a hierarchical approach might still not be able to return a solution in time. A decoupled approach, which is generally of a lower computational complexity, could then be used.

Selecting a Representation Class

Given an approach to combine the motion-planning algorithms (see the "Selecting an Approach to Combine Algorithm Classes" section), this section provides a method to select a suitable class of representations. This section discusses the robot representation and introduces the available classes to represent the environment: namely, an exact representation, an approximate representation, and a topological representation. The method is shown in Figure 2.

Representation Classes

To do motion planning, the robot, the workspace, and the obstacles therein need to be represented. The robot representation is determined by the configuration parameters that define the shape of the robot. The mobile robots considered in this article are rigid bodies moving on a plane. Hence, the workspace $\mathcal{W}$ of the robot is a two-dimensional (2-D) Euclidian space $\mathbb{R}^2$. The pose of the robot in this workspace can be uniquely defined by two position coordinates and one orientation coordinate. As a result, the configuration space $\mathcal{C}$ in which planning is performed typically has three dimensions. The configuration space can be extended to describe the rate of change of configurations, thereby forming a state space $\mathcal{S}$. To solve a motion-planning problem, a robot configuration is typically represented as a point in the configuration space. Obstacles are represented as configurations for which the robot is in collision.

The concept of the configuration space transforms the problem of planning the motion of an arbitrarily shaped robot into the problem of planning a sequence of configurations or states that move the robot toward the goal. To be able to search for a sequence of configurations or states in the solution space, the solution space must be represented in such a way that it connects configurations or states, i.e., the connectivity of the free space must be captured.

An explicit representation computes the obstacle region $\mathcal{C}_{\text{obs}}$ for all the obstacles. Implicit representations avoid this explicit construction and instead rely on a collision checking model to verify whether a configuration is free or not [4], [12]. In [4], this is called sampling-based motion planning. Note that, in this terminology, sampling does not necessarily imply stochastic sampling but refers to the configurations for which a collision check is performed. Two sampling methods exist: 1) deterministic and 2) stochastic or random.

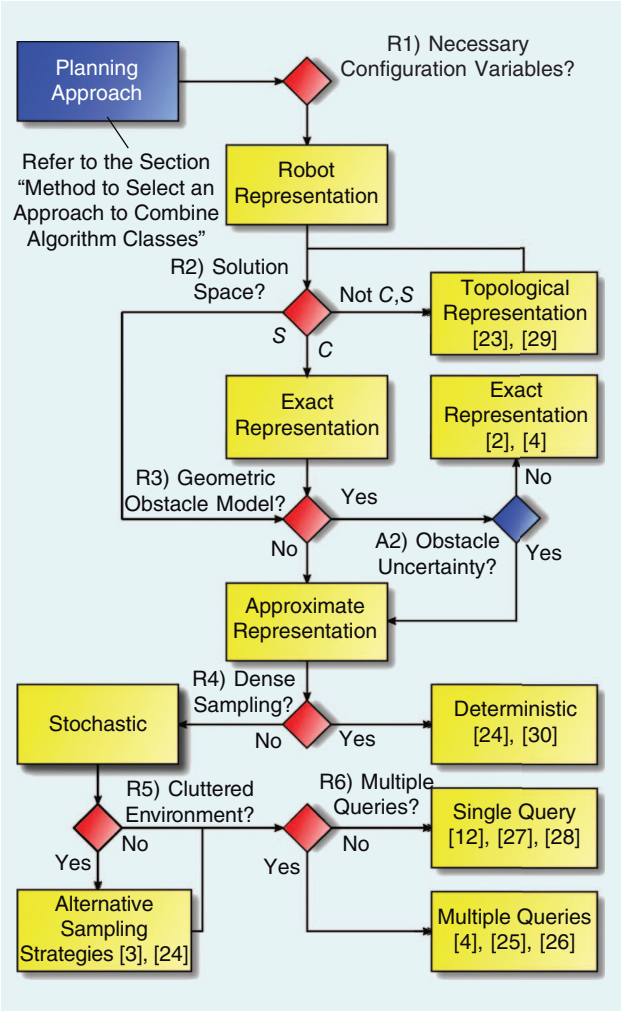

Figure 2. A method to arrive at an appropriate class of representations. The numbers R1–R6 (red) correspond to design criteria posed in the "Method to Select a Representation Class" section. Number A2 (blue) corresponds to a criterion posed in the "Method to Select an Approach to Combine Algorithm Classes" section.

Besides being explicit or implicit, a representation can be either exact or approximate. An exact representation requires obstacles to be described by a geometric model, e.g., a 2-D polygon or a three-dimensional (3-D) polyhedron. An exact representation generally captures the connectivity of the free space in a network of curves, called a road map [2]. An overview of algorithms that compute an exact representation is given in [2] and [4]. Most exact representations are limited to the configuration space $\mathcal{C}$ as, in the presence of kinodynamic constraints, they lose completeness [4]. On the other hand, approximate representations do allow the integration of kinodynamic constraints and can thus be used to represent a state space $\mathcal{S}$.

An approximate representation shows the free space and the occupied space as collections of cells, which have the same simple, prespecified shape, e.g., a hyper-rectangle. Due

to the approximate nature, only a weaker notion of completeness and optimality can be achieved, i.e., resolution completeness and optimality [4], but it is able to deal with arbitrary obstacle shapes.

A deterministic sampling method uses a grid that is placed over a robot's solution space. This resolution is complete and optimal, meaning the completeness and optimality depend on the resolution of the grid. The size of the grid grows exponentially with the dimension of the solution space [24]; hence, deterministic sampling is typically used for low-dimensional solution spaces. If desired, this low dimensionality enables the possibility to make the representation explicit, e.g., to inflate $\mathcal{C}_{\text{obs}}$ with the robot radius to obtain an explicit cost map. A stochastic sampling method relies on random sampling. A configuration is evaluated for collisions as it is sampled, and hence these methods are always implicit. An example of a representation based on random sampling is a probabilistic road map (PRM) [25]. There exist various variants of PRMs that are probabilistically complete, i.e., the probability that the planner returns a solution, if one exists, increases to one as the number of samples approaches infinity. More recent developments, such as PRM*, also approach asymptotical optimality [26].

A deterministic sampling method uses a grid that is placed over robot's solution space.

A single-query planner or incremental planner, such as a rapidly exploring random tree (RRT) [27], an expansive-space tree (EST) [12], and derivatives thereof, also relies on random sampling. However, instead of computing a connectivity graph and subsequently searching this, it interweaves representation and search. Similar to PRM*, recent developments, such as RRG, RRT* [26], and RRT# [28], also approach probabilistic completeness and asymptotical optimality.

A topological representation is a more abstract representation that describes the relations among the features of the environment without an absolute reference system [23], [29]. Such environmental features can be landmarks or identifiable locations such as a room of a house or an intersection of roads. A topological map, which results from a topological representation, has the advantage of being more compact and more stable with respect to sensor noise and uncertainty in a priori information on the world space than a metric map, which results from an exact or approximate representation. The topological map does not provide a framework to control the robot, since it does not consider the dynamics of the robot and cannot capture the moving obstacles. Therefore, another planner is required in a hierarchical approach, as mentioned in the "Approaches to Combine Algorithm Classes" section.

In the case of the reactive planners in the "Approaches to Combine Algorithm Classes" section and the single-query planners in this section, representation and search are tightly coupled, and it can, therefore, be difficult or impossible to make a clear distinction between these two steps, as is done in [1]. However, for many other motion planners, this distinction can be made, and since this is beneficial to the clarity of the motion-planning method, it will be maintained throughout this article. Although one might argue that single-query planners are search algorithms rather than representations, they are already introduced in this section to make sure that single-query planners are not discarded in the method before proceeding to the selection of search algorithm classes.

Method to Select a Representation Class

Given an approach to combine the motion-planning algorithms, the following design criteria, which are numbered R1–R6, form a method to select a suitable class of representations. The method is visualized in Figure 2.

R1) What configuration variables are necessary to describe the robot? Three configuration variables are required to uniquely define a robot's position and orientation on the ground plane. Not all the variables might be necessary to describe the robot, and hence the computational complexity of finding a solution in the resulting configuration space $\mathcal{C}$ can be reduced. For example, it is a common practice to consider only the position of the robot when planning, thereby omitting its orientation and limiting the dimension of $\mathcal{C}$ [2]. This can, however, result in incorrectness, e.g., a robot with an elongated shape that is described by a circular shape may not fit through a narrow passage. The greater the approximation of the shape is, the more likely the motion planner will be incorrect. This also depends on the smallest passage width or, generally, the structure of the environment.

R2) What is the solution space? The selection of an approach to combine the motion-planning algorithms in the "Selecting an Approach to Combine Algorithm Classes" section results in a solution space that needs to be represented and searched for a solution. An exact representation is an option if the solution space is $\mathcal{C}$. An approximate method is typically used if the solution space is $\mathcal{S}$, as mentioned in the "Representation Classes" section. A topological representation can be used if the environment can be represented as a collection of connected locations or features. A topological representation reduces the computational complexity, as it is more compact than a metrical representation such as an exact or approximate representation. A planner that uses a topological representation requires a succeeding planner that uses a metrical representation, as a robot is typically controlled in a metrical frame.

R3) Can obstacles be described by a geometric model? An exact representation method is complete and can provide an optimal solution if the solution space is $\mathcal{C}$ and obstacles can by modeled by a geometric model [1], [2]. This requires complete knowledge of obstacle shapes; thus, any uncertainty, unless bounded, is prohibitive. Otherwise, an approximate representation is preferred.

R4) Can a solution be obtained in time with dense sampling? A deterministic method samples a space using a regular grid. An approach is generally close to completeness, and its solutions are close to optimality when dense sampling is used. The

computational complexity of a deterministic sampling method can be too high to obtain a solution within a certain time requirement. Therefore, this method is typically limited to low-dimensional solution spaces as the size of the grid grows exponentially with the dimension of the solution space [24]. A multiresolution deterministic sampling representation, such as a quadtree (2-D) or an octree (3-D) [30], is an option if the computational complexity is limiting. As this representation can represent volumes with a lower resolution, it is typically more efficient for planning purposes than a regular, deterministic representation. A random sampling method is even of a lower complexity, as it generally requires fewer samples to represent the world space than a deterministic sampling method. As a result, it can be applied to higher-dimensional solution spaces.

R5) Is the environment cluttered? In a cluttered environment, care needs to be taken to prevent the incompleteness when using an approximate representation method. The resolution of sampling must be dense enough to allow the robot to pass through the smallest clearance between obstacles. In particular, for random sampling methods, it is important that the used sampling strategy ensures dense enough sampling in small passages. This difficulty that the sampling-based planners face, known as the "narrow passage problem" [31], can be overcome using different sampling strategies [3], [24].

R6) Can the planner be used for multiple queries? It can be beneficial to use a road map representation if a motion planner must generate multiple solutions within the same world space. This can be an explicit representation, such as a Voronoi diagram or visibility graph [4], but also an implicit variant, such as a PRM [25] or PRM* [26]. A road map only needs to be computed once and can subsequently be used for multiple searches; hence, the computational complexity of the representation is limited to a precomputation step. A road map can be recomputed or updated if the environment changes. On the other hand, if only one solution is required, it is not necessary to generate a complete representation of the world space. Representation and searching could then be interleaved using a single-query planner such as an RRT [27], EST [12], RRT* [26], or RRT# [28]. Note that these algorithms do not require a search method as is introduced in the "Selecting a Search Algorithm Class" section.

Selecting a Search Algorithm Class

The sections "Selecting an Approach to Combine Algorithm Classes" and "Selecting a Representation Class" discuss the available approaches to combine the motion-planning algorithms and the available classes of representation algorithms. In this section, two available classes of search algorithms are discussed: 1) uninformed and 2) informed. Depending on the designer's requirements, these classes of algorithms can be combined with additional features, such as incremental replanning and anytime behavior. A method, shown in Figure 3, is provided to select a suitable class of search algorithms, given an approach to combine the motion-planning algorithms (see "Selecting an Approach to Combine Algorithm Classes" section).

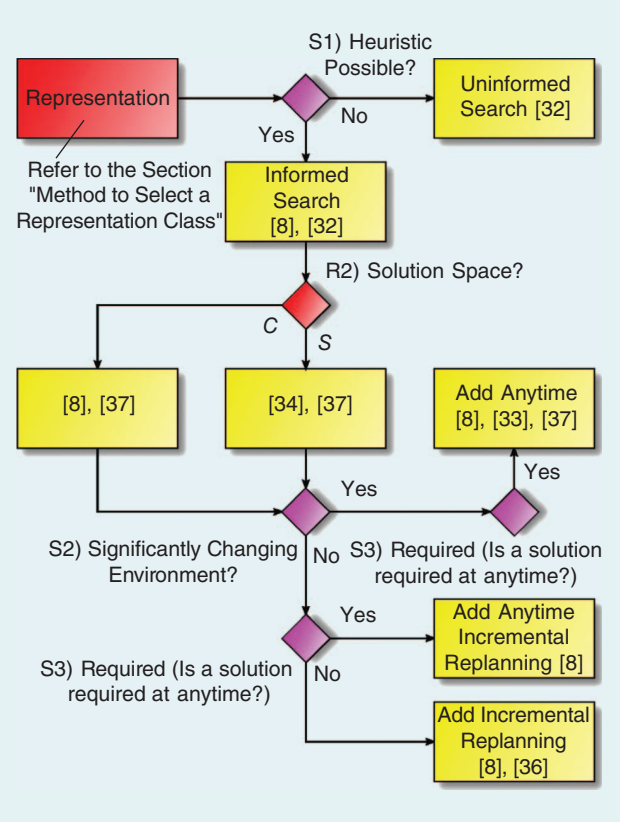

Figure 3. A method to arrive at an appropriate class of search algorithms. The numbers S1–S3 (purple) correspond to the design criteria posed in the "Method to Select a Search Algorithm Class" section, and R2 (red) corresponds to a criterion posed in the "Selecting a Representation Class" section.

Search Algorithm Classes

Most representations transform the continuous problem of finding a solution in the free space into a discrete problem of searching a graph with the nodes that represent the configurations or states of the robot. Therefore, uninformed and informed search algorithms can be applied.

An uninformed search algorithm moves through the graph without any preference for the location of the goal. Well-known algorithms are the breadth-first search algorithm and depth-first search algorithm. An optimal path with respect to a specified cost criterion can be achieved using Dijkstra's algorithm [32].

If the direction of the goal node is known, the search can be directed, i.e., it can be biased toward the goal. A search algorithm that includes information about the goal is called an informed search algorithm. To achieve this, a heuristic is formulated, i.e., a function of nodes that hypothesizes cost toward the goal node. The choice for the next node to explore is then based on this heuristic cost, e.g., defined as the Euclidean distance toward the goal. The most well-known algorithm is the A* algorithm. Similar to the uninformed algorithms, A* also returns the optimal path, but the search typically is of

lower computational complexity, i.e., it explores less space and, thus, returns a solution faster [32]. To ensure completeness and optimality, a heuristic must satisfy two properties: 1) consistency and 2) admissibility [32]. A heuristic that satisfies these two properties never overestimates the cost of reaching the goal.

Method to Select a Search Algorithm Class

Given an approach to combine motion-planning algorithms, the following design criteria, numbered S1–S3, form a method to select a suitable class of search algorithms. The method is shown in Figure 3.

S1) Can a heuristic be defined to speed up the search? The use of a heuristic to the goal can reduce the complexity of a search [8]. An informed search is preferred over an uninformed search, if a consistent and admissible heuristic [32] can be defined. In particular, for a search in a high-dimensional solution space, a heuristic can greatly reduce the complexity. A heuristic is then typically informed with more than just the distance to the goal by using a search in a lower dimension as a heuristic to inform a search in the full dimension of the solution space [33], [34]. For example, a planner that does a search in the state space $\mathcal{S}$ can be informed with a heuristic that does a search in the configuration space $\mathcal{C}$. This heuristic search ignores robot constraints and moving obstacles, but it can reduce the search space of $\mathcal{S}$ substantially, and, thus, it can return a solution faster. The reduction in the space depends on whether the heuristic solution is expected to be close to the optimal solution.

The single-query planners introduced in the "Selecting a Representation Class" section can also benefit from adding a goal bias or other heuristic to the sampling algorithm. Examples of such approaches can be found in [35].

S2) Does the environment change significantly between consecutive searches? A plan can become infeasible during execution because of appearing or moving obstacles. Therefore, a motion planner can adopt a replanning strategy. Replanning can be performed by computing an entirely new plan. One could also use incremental replanning using the results of previous searches to generate a new plan faster [36]. The benefit of considering previous solutions depends on the environment. If there is no significant change, a replan can be acquired faster, but when the results of previous solutions are less useful, it can even be disadvantageous to the task of arriving at a new solution [8].

S3) Is a solution required at anytime? It is desirable to achieve a solution as fast as possible. However, this solution is typically not optimal. An anytime search algorithm can return a solution within a certain time constraint and with a certain bound on its suboptimality [8]. As time allows, this bound is tightened, and, given enough time, the optimal solution is returned [37]. An optimal search algorithm typically acquires this solution faster, but an anytime algorithm allows a robot to start executing an initial suboptimal solution and improve this solution along the way; examples can also be found in [26].

This concludes this discussion of the motion-planning method. An overview of the requirements and the corresponding design criteria is presented in Table 1. This overview shows that several design criteria involve multiple requirements. This clearly indicates the tradeoffs that are commonly present when designing a navigation system, e.g., one could try to design a system that returns optimal solutions in the presence of moving obstacles, but this would be computationally extremely complex. More of these tradeoffs can be derived from Table 1.

Table 1. The design requirements and the corresponding selection criteria of the method.

Requirement	Design Criterium
Environment	
Known/unknown	A1, A6
Uncertainty in obstacle representation	A2, A4, A6
Moving obstacles	A3, A4, A6, S2
Robot model	
Correctness	R1
Kinodynamic constraints	A4, A5, A6, A7, R2
Application	
Completeness	A2, A4, A5, R1, R3, R4, R5
Optimality	A3, A4, A5, R3, R4, S3
Computational complexity	A3, A4, A5, A6, A7, R4, R6, S1

Method Verification

Three use cases from the literature involving the application of a motion planner in practice are evaluated to verify the proposed method: 1) the PR2 service robot that autonomously navigated a marathon distance of 42.15 km in a real office environment [13], 2) the autonomous vehicle Boss that won the U.S. Defense Advanced Research Projects Agency Urban Challenge [38], and 3) a flexible automated guided vehicle (AGV) in partially structured warehouses [39].

Application of the Method to Use Case I

The PR2 service robot has an omnidirectional base platform with a top speed of 1 m/s. It navigates an office environment that is cluttered with a variety of arbitrarily shaped obstacles [13].

Selecting an Approach to Combine Algorithm Classes

The office environment is only partly known, so the robot must replan as it receives new information about the world (A1). The depth accuracy of the sensors that are used for navigation is 1.5 cm. Hence, the resulting uncertainty must be captured in a bounded uncertainty region (A2). The office environment is populated by people, so the environment contains moving obstacles (A3). The trajectories of people are unpredictable (A4), and the world space is only partially known. Therefore, the representation of the world space will change rapidly. As the robot is desired to navigate at a reasonable speed of about 0.5 m/s, a reactive approach is the most suitable approach to avoid collision in real-time (A5 and A6). A reactive approach requires an

obstacle avoidance algorithm and a global planner (see "Approaches to Combine Algorithm Classes" section).

Selecting a Representation Class

With only 5-cm clearance on both sides when traversing a typical doorway, a circular approximation of the PR2 will result in incompleteness. Thus, the robot is best described by a rigid body (R1). The solution space consists of multiple rooms and corridors and, therefore, lends itself for a topological representation. This representation reduces the size of the solution space of the global planner, which is equal to C (R2). An approximate representation method is preferred over an exact representation method as the obstacles are of arbitrary shape and their representation is uncertain (R3). The environment is cluttered with small obstacles, such as office supplies; thus, dense sampling is required to prevent incompleteness. A deterministic sampling method is, therefore, preferred over a stochastic sampling method (R4). A multiresolution representation is advised to negotiate the computational complexity as an office environment has large open space but also consists of cluttered occupied spaces.

Selecting a Search Algorithm Class

The PR2 is supplied with a navigation goal. The direction of this goal can be used as a heuristic to speed up the search (S1). In a cluttered office environment, a replan can be significantly different from an original search, and, thus, recomputing an entire new plan is preferred (S2). It is not required for a robot to obtain a solution immediately upon a goal request, but it is desirable. Therefore, the search algorithm can be extended with an anytime algorithm (S3).

Comparison of the Method's Outcome and the Use Case

Similar to the prescription by the method, the PR2 motion planner [13] operates with a reactive approach as it uses a global planner and a local obstacle avoidance algorithm [20]. The local algorithm uses a rigid body description, but the robot's shape is approximated by an inscribed circle in the global representation of the world space. This can result in incompleteness as the global planner may produce a path through a narrow passage, e.g., a half-opened door that the local planner cannot execute. We identified this as a limitation [13]. As the method prescribes, this can be prevented if the global planner would also use a rigid body description of the robot. A motion planner for the PR2 that does this is presented in [40]. The PR2 motion planner does not use a topological representation on top of the global configuration space representation, e.g., as used in [23]. This is, however, identified for a future work. Furthermore, the global planner uses an informed search algorithm (A*) like the method indicated, but it is not an anytime algorithm.

Application of the Method to Use Case II

The autonomous driving vehicle Boss uses two different planners [38]: one during nominal on-road driving and one during more unstructured driving, e.g., in a parking lot or during error recovery. The latter planner is used for the verification of the method.

Selecting an Approach to Combine Algorithm Classes

Replanning is required, as Boss faces unknown spaces as it navigates (A1). Boss' sensors have a limited resolution, and thus a bounded uncertainty region is necessary (A2). During navigation, the vehicle encounters other vehicles (A3) whose future trajectories are predictable in short term (A4); hence, a time dimension can be added for a time-optimal solution. A solution in the form of a plan in the full-dimensional solution space is necessary to be time optimal, but as vehicles are only predictable in the short term, it is only necessary to consider a time dimension for that short term (A5). A plan is necessary to consider the kinematic constraints of the vehicle (A6). To achieve a local optimal solution, a hierarchical approach is suggested (A7).

Selecting a Representation Class

A vehicle is subject to kinematic constraints that must be considered while planning (R1). The velocity also needs to be considered during planning to generate feasible motions in the presence of obstacles, resulting in S as a solution space (R2). Given the dimension and size of the solution space, a stochastic sampling method is suitable (R4). Sampling is required to be dense (R5), especially near the goal configuration as this is typically in a narrow part of the environment such as a parking lot. A single-query planner suits as previous solutions are not reusable (R6).

Selecting a Search Algorithm Class

The vehicle is supplied with a navigation goal that can be used as a heuristic (S1). Incrementally updating the solution as new information is acquired can be beneficial, as a traffic environment, in general, is not very cluttered (S2). To prevent the vehicle from stopping to calculate a plan, it is important that it returns a solution within a certain time constraint. Hence, an incremental, anytime replanning algorithm is necessary (S3).

Comparison of the Method's Outcome and the Use Case

Boss uses a hierarchical approach [33], which is also prescribed by the method. The approach includes a global planner that searches for a feasible motion plan in four-dimensional (4-D) (x, y, θ, v) and a local motion planner operating in the same space, which selects a trajectory that follows the global plan. Short, feasible maneuvers are computed offline to substantially reduce the time to acquire a solution. These maneuvers are concatenated at planning time in a single-query fashion to obtain a motion plan. In the vicinity of the vehicle and the goal, the search is dense, like the method indicated, while in other areas, the set of possible maneuvers is more coarse. A combination of heuristics is used to further reduce the time to acquire a solution. Similar to the method's prescription, an incremental, anytime algorithm is used. In contrast to the method, a time dimension is not explicitly incorporated to deal with other moving vehicles. This is solved by marking the space that another

vehicle will occupy over a short-term prediction as occupied in a static map. However, this can result in suboptimal behavior in the presence of moving vehicles, but it returns a solution faster, as the search space does not include a time dimension.

Application of the Method to Use Case III

Contrary to many AGVs, the fork-lift truck considered [39] incorporates a high degree of on-board autonomy to be flexible with respect to floor layout changes and to decrease the amount of manual work when establishing the a priori knowledge of the workplace.

Selecting an Approach to Combine Algorithm Classes

Replanning is required due to the presence of unknown and possibly dynamic obstacles such as other vehicles (A1). Again, a finite sensor resolution demands a bounded uncertainty region (A2). The environment does contain moving obstacles (A3), but there is no prediction of their velocity or future position (A4). It is not necessary to plan in the full state space S (A5), but a plan in the reduced space is necessary to get as close as possible to optimality (A6). Due to the presence of kinematic constraints, a hierarchical approach is necessary (A7). In the following, the path planner will be discussed.

Selecting a Representation Algorithm Class

The robot pose is described by three coordinates. The hierarchical approach, however, permits the omission of the orientation when computing the path toward the goal, since the kinodynamic constraints will be accounted for by the local planner (R1). A topological representation is desirable due to the size of the environment (R2). Due to the presence of unknown obstacles, an approximate representation is required (R3). Since the planner only needs to work in 2-D and the size is limited due to the presence of a topological representation, a grid-based planner can be used (R4).

Selecting a Search Algorithm Class

The planner plans toward the subgoals of the topological map. Hence, a goal bias can be used to speed up the search (S1). As mentioned before, replanning is required due to the changes in the environment (S2). It is desired to plan in real time during navigation, so an anytime incremental planner would be most suitable (S3).

Comparison of the Method's Outcome and the Use Case

Similar to the prescription of the method, the navigation system in [39] combines a topological representation with a 2-D grid map and a local planner. The topological representation in [39] even consists of two levels: 1) each node represents a zone, typically a roomlike space and 2) the nodes represent doors, wait points, dock points, and way points. The grid representation is searched by a D* algorithm with a goal bias, since this resulted in low replanning times. The initial search, however, did not meet the real-time requirement; hence, it cannot be considered an anytime planner. The local planner consists of three separate behaviors: 1) an obstacle avoider, 2) path tracker, and 3) dock driver. The contributions of the behaviors, which might be executed concurrently, are fused. The obstacle avoidance relates to the potential field method, while the path tracking is an implementation of the receding horizon control.

Conclusions and Future Work

This article presents a method to select a representation class, a search algorithm class, and an appropriate approach to combine these for a mobile robot that moves on a ground plane. In the first part, an appropriate approach to combine the algorithm classes is selected. In the following two parts, a representation class and a search algorithm class to accomplish the two steps of motion planning are selected. An overview of the requirements and the corresponding design criteria are presented in Table 1. Considering both the steps of a motion planner, for all the requirements as introduced in the "Motion Planning" section, the method can be used to find an appropriate combination of motion-planning algorithm classes.

The method is verified with three use cases where the existing motion planners are successfully applied in practice. The method's outcome resembles the existing motion planners, and the recommendations from the method reveal the points for improvement that are indicated by the original authors. It can, therefore, be concluded that the method is beneficial to the selection of an appropriate motion-planning approach and representations and search algorithm classes. To verify the method more extensively, future work will apply the method to more robots, which already feature a motion planner that functions well in practice. As a part of this, we will apply the plan to the robots of the team Tech United Eindhoven (http://www.techunited.nl/) and evaluate the resulting planner designs in the RoboCup competitions. Furthermore, to increase the practical usability, we are also interested in complementing this work with a tool that contains a database of the selected references and queries the decision trees of the method for an appropriate combination of motion-planning algorithms.

References

[1] Y. Hwang and N. Ahuja, "Gross motion planning—A survey," *ACM Comput. Surveys*, vol. 24, no. 3, pp. 219–291, 1992.

[2] J. Latombe, *Robot Motion Planning*. Berlin, Germany: Springer-Verlag, Dec. 1990.

[3] H. Choset, *Principles of robot motion: theory, algorithms, and implementation*. Cambridge, MA: MIT Press, 2005.

[4] S. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[5] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, 1986.

[6] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *J. Intell. Robot. Syst.*, vol. 57, no. 1–4, pp. 65–100, 2010.

[7] J. F. Canny, *Complexity of Robot Motion Planning*. Cambridge, MA: MIT press, 1988.
[8] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Proc. Workshop Planning Uncertainty Autonomous Systems Int. Conf. Automated Planning Scheduling*, 2005.
[9] J. Giesbrecht, "Global path planning for unmanned ground vehicles," in "Defence R&D Canada–Suffield," Tech. Rep. DRDC-Suffield-TM 2004-272, 2004.
[10] J. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," in *Robot Motion Planning and Control*, Berlin, Germany: Springer-Verlag, 1998, pp. 1–53.
[11] H. Choset, "Coverage for robotics—A survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, no. 1, pp. 113–126, 2001.
[12] D. Hsu, R. Kindel, J. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. Robot. Res.*, vol. 21, no. 3, pp. 233–255, 2002.
[13] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: robust navigation in an indoor office environment," in *Proc. IEEE Int. Conf. Robotics Automation*, 2010, pp. 300–307.
[14] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *J. ACM*, vol. 40, no. 5, pp. 1048–1066, 1993.
[15] T. Fraichard, "Trajectory planning in a dynamic workspace: A 'state-time space' approach," *Adv. Robot.*, vol. 13, no. 1, pp. 75–94, 1998.
[16] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robotics Automation*, 2011, pp. 4569–4574.
[17] H. Zhang, J. Butzke, and M. Likhachev, "Combining global and local planning with guarantees on completeness," in *Proc. IEEE Int. Conf. Robotics Automation*, 2012, pp. 4500–4506.
[18] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, p. 90, 1986.
[19] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.
[20] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Automat. Mag.*, vol. 4, no. 1, pp. 23–33, 1997.
[21] J. Minguez, L. Montano, T. Simeon, and R. Alami, "Global nearness diagram navigation (gnd)," in *Proc. IEEE Int. Conf. Robotics Automation*, 2001, vol. 1, pp. 33–39.
[22] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *Proc. IEEE Int. Conf. Robotics Automation*, 1993, vol. 2, pp. 802–807.
[23] P. Zavlangas and S. Tzafestas, "Integration of topological and metric maps for indoor mobile robot path planning and navigation," *Methods Appl. Artif. Intell.*, vol. 2308, ISSN: 0302-9743, pp. 746–746, 2002.
[24] S. R. Lindemann and S. M. LaValle, "Current issues in sampling-based motion planning," in *Robotics Research* (Springer Tracts in Advanced Robotics, no. 15). Berlin, Germany: Springer-Verlag, 2005, pp. 36–54.
[25] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, 1996.
[26] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
[27] S. LaValle and J. Kuffner, Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.
[28] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *Proc. IEEE Int. Conf. Robotics Automation*, 2013, pp. 2421–2428.
[29] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli, "Local metrical and global topological maps in the hybrid spatial semantic hierarchy," in *Proc. IEEE Int. Conf. Robotics Automation*, 2004, vol. 5, pp. 4845–4851.
[30] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robot.*, vol. 34, no. 3, pp. 189–206, 2013.
[31] D. Hsu, L. Kavraki, J. Latombe, R. Motwani, S. Sorkin, "On finding narrow passages with probabilistic road map planners," in *Proc. Int. Workshop Algorithmic Foundations Robotics*, 1998, pp. 141–154.
[32] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall, 2010.
[33] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime search in dynamic graphs," *Artif. Intell.*, vol. 172, no. 14, pp. 1613–1643, 2008.
[34] M. Phillips and M. Likhachev, "SIPP: Safe interval path planning for dynamic environments," in *Proc. IEEE Int. Conf. Robotics Automation*, 2011, pp. 5628–5635.
[35] A. Qureshi, S. Mumtaz, K. Iqbal, B. Ali, Y. Ayaz, F. Ahmed, M. Muhammad, O. Hasan, W. Y. Kim, and M. Ra, "Adaptive potential guided directional-RRT," in *Proc. IEEE Int. Conf. Robotics Biomimetics*, 2013, pp. 1887–1892.
[36] S. Koenig, M. Likhachev, Y. Liu, and D. Furcy, "Incremental heuristic search in AI," *AI Mag.*, vol. 25, no. 2, p. 99, 2004.
[37] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* search with provable bounds on suboptimality", in *Proc. Conf Neural Information Processing Systems (NIPS)*, S. Thrun, L. Saul and B. Schölkopf, Eds., Cambridge, MA: MIT Press, 2003.
[38] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot.*, vol. 25, no. 8, pp. 425–466, 2008.
[39] H. Martínez-Barberá and D. Herrero-Pérez, "Autonomous navigation of an automated guided vehicle in industrial environments," *Robot. Comput.-Integr. Manufact.*, vol. 26, no. 4, pp. 296–311, Aug. 2010.
[40] A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, and S. Chitta, "Navigation in three-dimensional cluttered environments for mobile manipulation," in *Proc. IEEE Int. Conf. Robotics Automation*, 2012, pp. 423–429.

Janno Johan Maria Lunenburg, Eindhoven University of Technology, The Netherlands. E-mail: j.j.m.lunenburg@tue.nl.

Sebastiaan Antonius Maria Coenen, Eindhoven University of Technology, The Netherlands. E-mail: s.a.m.coenen@student.tue.nl, coenen.bas@gmail.com.

Gerrit Naus, Eindhoven University of Technology, The Netherlands. E-mail: g.j.l.naus@tue.nl.

Marinus Jacobus Gerardus van de Molengraft, Eindhoven University of Technology, The Netherlands. E-mail: m.j.g.v.d.molengraft@tue.nl.

Maarten Steinbuch, Eindhoven University of Technology, The Netherlands. E-mail: m.steinbuch@tue.nl.