sensors

MDPI

*Article*

# Navigation of Multiple Disk-Shaped Robots with Independent Goals within Obstacle-Cluttered Environments

**Panagiotis Vlantis [1], Charalampos P. Bechlioulis [1,*] and Kostas J. Kyriakopoulos [2]**

[1] Department of Electrical and Computer Engineering, University of Patras, University Campus, 26504 Rion, Greece
[2] School of Mechanical Engineering, National Technical University of Athens, 15780 Athens, Greece
* Correspondence: chmpechl@upatras.gr

**Abstract:** In this work, we propose a hybrid control scheme to address the navigation problem for a team of disk-shaped robotic platforms operating within an obstacle-cluttered planar workspace. Given an initial and a desired configuration of the system, we devise a hierarchical cell decomposition methodology which is able to determine which regions of the configuration space need to be further subdivided at each iteration, thus avoiding redundant cell expansions. Furthermore, given a sequence of free configuration space cells with an arbitrary connectedness and shape, we employ harmonic transformations and harmonic potential fields to accomplish safe transitions between adjacent cells, thus ensuring almost-global convergence to the desired configuration. Finally, we present the comparative simulation results that demonstrate the efficacy of the proposed control scheme and its superiority in terms of complexity while yielding a satisfactory performance without incorporating optimization in the selection of the paths.

**Keywords:** multi-robot navigation; motion planning; cell decomposition

check for updates

**Citation:** Vlantis, P.; Bechlioulis, C.P.; Kyriakopoulos, K.J. Navigation of Multiple Disk-Shaped Robots with Independent Goals within Obstacle-Cluttered Environments. *Sensors* **2023**, *23*, 221. https://doi.org/10.3390/s23010221

Academic Editors: Nir Shvalb and Amir Shapiro

Received: 21 October 2022
Revised: 19 December 2022
Accepted: 21 December 2022
Published: 25 December 2022

## 1. Introduction

The autonomous operation of robotic platforms inside cluttered environments constitutes an actively studied research topic, with autonomous navigation undeniably being a fundamental aspect of it. Additionally, as the tasks that robots are entrusted with grow in complexity, the employment of multi-robot systems, which generally exhibit a higher robustness and versatility than their single-robot alternatives, progressively increases. Thus, the multi-robot navigation problems to be addressed become more challenging day by day, raising the need for more efficient path and motion planning schemes.

Several methodologies can be found in the literature for coordinating the motion of two or more robots such that a desired configuration is reached under standard collision avoidance specifications. Combinatorial and algebraic approaches, such as the ones presented in [1,2], were among the earliest to be considered. However, despite their elegance, their complexity renders them impractical for addressing cases with non-trivial sizes of robotic teams. On the other side, probabilistic sampling methods, such as Rapidly Exploring Random Trees [3] and Probabilistic Roadmaps [4], constitute a popular solution employed in the recent literature due to their simplicity and their ability to efficiently handle large configuration spaces while being subjected to a variety of constraints. Nevertheless, these methodologies are generally having a hard time addressing problems with constricted configuration spaces (e.g., workspaces densely occupied by obstacles or narrow corridors). To alleviate these issues, various attempts were made. In [5], a novel sensory steering algorithm was designed which used the local Voronoi decomposition of the workspace to significantly improve the path-planning performance of sampling-based algorithms near difficult regions, such as narrow passages. In [6,7], the authors proposed a scheme that samples entire manifolds instead of isolated configurations, which are, in turn, used for approximating the configuration space's connectivity graph, thus allowing the planner to perform significantly better even in tight workspaces. Alternatively, when a common graph representation of the workspace is shared among the agents, efficient methodologies for coordinating their transitions were proposed in [8–12]. On the other hand, a methodology was presented in [13,14], which addresses cases where the motion of each robot is restricted to a distinct graph by building a composite roadmap (i.e., the Cartesian product of the individual graphs). Furthermore, more efficient extensions of this approach, which work on implicitly defined composite roadmaps and, potentially, lower-dimensional configuration spaces, can be found in [15–18]. The approximate cell decomposition [19] and Slice Projection [20–22] constitute alternative methodologies which were successfully employed for tackling robot navigation problems with complex configuration spaces [23–25] and generally exhibit fast exploration capabilities when coupled with hierarchical adaptive subdivision schemes guided by suitable heuristics. For more details on the related literature, the reader may refer to the following recent comprehensive review paper [26].

In this work, we address the navigation problem for a team of disk-shaped robots that operate within an arbitrary, obstacle-cluttered, planar and connected workspace (see Figure 1). Given an initial and desired configuration for the robotic team, we design a high-level hierarchical cell decomposition planner which is tasked with the exploration of the system's configuration space to discover a sequence of cells that the robots can safely traverse toward the desired configurations. One of the strong points of the proposed algorithm is the use of a suitable labeling mechanism for selecting the regions of the configuration space to be subdivided at each iteration. Particularly, by computing an over- and an under-approximation of each robot's footprint, our algorithm can determine which cells may contain feasible configurations of the system while automatically discarding the cells that are determined to contain none. Finally, having obtained a sequence of traversable cells, we equip each robot with decentralized low-level control laws based on harmonic maps and adaptive harmonic potential fields, originally introduced in [27], which guarantee the safe and almost-global convergence to the goal. It should be noted that the high-level planner produces a series of cells through which the multi-robot system has to go in order to attain the final desired configuration. The aforementioned low-level controller, which was extracted by our previous work [27], simply guarantees a safe transition between successive cells. Thus, any other motion planning algorithm would also suffice.

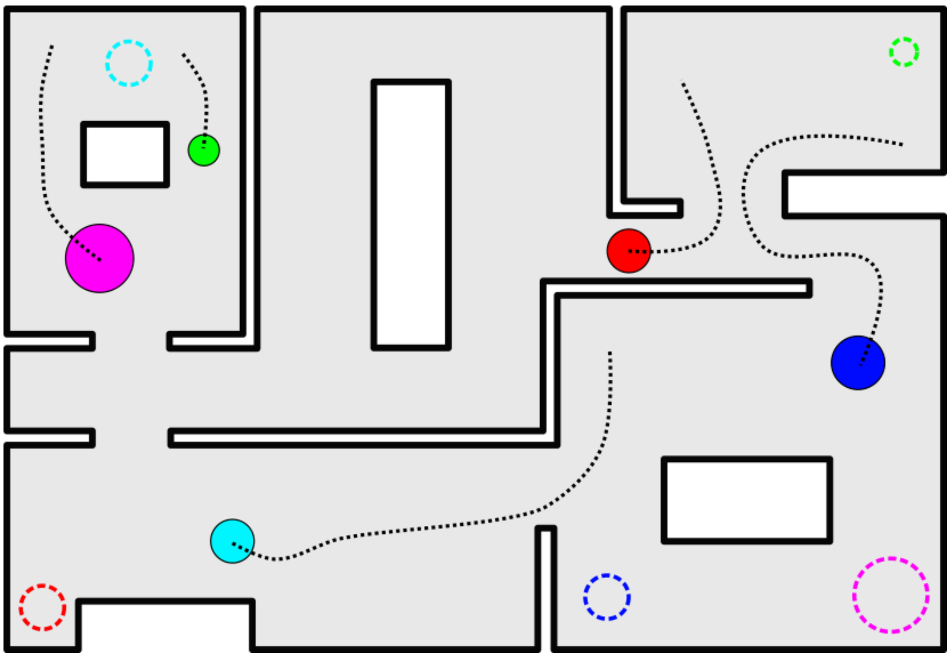


**Figure 1.** Multiple robots (solid colored disks) navigating to their corresponding goal configurations (dashed circles).

The contribution of this work is the adaptive subdivision algorithm for obtaining a sequence of traversable cells that connect the given initial and final configurations or determining that no solution exists. Unlike standard grid/cell-based methods, the proposed algorithm subdivides the cells adaptively, requiring no selection of some arbitrary resolution by the user. Moreover, unlike probabilistic sampling methods, the proposed method does not generate paths of configuration but rather sequences of safe sets, allowing the user to make use of a larger set of controllers in addition to standard trajectory tracking schemes. Additionally and contrary to the probabilistic sampling methods, which are inherently incapable of determining the infeasibility of a given problem (such algorithms require the user to specify an arbitrary upper bound of samples after which the algorithms abort) as a direct consequence of their probabilistic completeness nature, our algorithm partitions the configuration space into cells (dense subsets) of the configuration space and is capable of determining when no admissible path exists in finite time (see [19]).

The outline of this article is as follows. At the end of this section, we define some preliminary notions and the notation used throughout this work. In Section 2, we formulate the problem addressed in this work. In Section 3, we elaborate on the proposed planner's design as well as the velocity control scheme employed for safely executing the computed plan. Finally, the simulation results verifying the efficacy of the proposed control scheme are presented in Section 4.

*Notation:* Throughout this work, we shall use $\mathfrak{I}_N \triangleq \{1, 2, \ldots, N\}$ (resp., $\mathfrak{I}^{\star}_N \triangleq \{0\} \cup \mathfrak{I}_N$) to denote the set consisting of all natural numbers up to $N$, starting from 1 (resp., 0). Additionally, given sets $A$ and $B$, we use $\partial A$, $\text{int}(A)$ and $\text{cl}(A)$ to denote the boundary, interior and closure of $A$, respectively, and $A \setminus B$ to denote the complement of $B$ with respect to $A$.

## 2. Problem Formulation

We consider a team of $N_{\mathcal{R}}$ robots operating within a compact planar workspace $\mathcal{W} \subseteq \mathbb{R}^2$ occupied by a set of $N_o$ disjoint, fixed inner obstacles $\mathcal{O}_i, i \in \mathfrak{I}_{N_o}$. We assume that each robot $i$ has a disk-shaped body $\mathcal{R}_i \subset \mathbb{R}^2$ with radius $r_i > 0$. Let $\mathcal{F}_w$ and $\mathcal{F}_i, i \in \mathfrak{I}_{N_{\mathcal{R}}}$ be the coordinate frames arbitrarily embedded in $\mathcal{W}$ and $\mathcal{R}_i, i \in \mathfrak{I}_{N_{\mathcal{R}}}$, respectively. We shall refer to the origin of each $\mathcal{F}_i, i \in \mathfrak{I}_{N_{\mathcal{R}}}$ as the reference point of the corresponding robot. Moreover, without loss of generality, we assume that the reference point of each robot coincides with the center of its body. Let $p_i \triangleq \left[x_i, y_i\right]^T \in \mathbb{R}^2$ denote the relative position of $i$-th robot's reference point with respect to the workspace's coordinate frame $\mathcal{F}_w$, and let $\mathcal{R}_i(p)$ denote its footprint, i.e., the space occupied by robot $i$ when placed at position $p$. Throughout this work, we shall use $\mathfrak{C} \subseteq \mathbb{R}^{2N_{\mathcal{R}}}$ to denote the robotic system's configuration space and $P \triangleq \left[p_1^T, p_2^T, \ldots, p_{N_{\mathcal{R}}}^T\right]^T \in \mathfrak{C}$ to denote the stacked vector of robot positions. For the sake of brevity, we shall also use $P[i]$ to denote the $i$-th component of $P$, i.e., $P[i] = p_i$. Let $\mathcal{W}^o$ denote the complement of $\mathcal{W}$, i.e., $\mathcal{W}^o \triangleq \mathbb{R}^2 \setminus \mathcal{W}$. We also define a configuration $P$ as feasible iff the following conditions hold:

$$\begin{aligned} \mathcal{R}_i(P[i]) \cap \mathcal{R}_j(P[j]) &= \emptyset, \quad \forall i \neq j \in \mathfrak{I}_{N_{\mathcal{R}}} \\ \mathcal{R}_i(P[i]) \cap \mathcal{W}^o &= \emptyset, \quad \forall i \in \mathfrak{I}_{N_{\mathcal{R}}} \end{aligned} \tag{1}$$

and we shall use $\mathfrak{C}^f \subset \mathfrak{C}$ to denote the set of all feasible configurations of the robotic system, whereas its complement $\mathfrak{C}^o \triangleq \mathfrak{C} \setminus \mathfrak{C}^f$ corresponds to the set all infeasible configurations. Finally, we assume that the motion of each robot $i$ obeys the single-integrator kinematic model:

$$\dot{p}_i = u_i, \; i \in \mathfrak{I}_{N_{\mathcal{R}}} \tag{2}$$

where $u_i$ denotes the control input.

**Problem:** *Let $P_{\text{init}}$ and $P_{\text{des}}$ be two given feasible configurations of the multi-robot system that belong to the same segment of $\mathfrak{C}^f$. Our goal is to design a control scheme that drives any robot $i$,* *initialized at $p_{\text{init},i} = P_{\text{init}}[i]$, to the specified desired position $p_{\text{des},i} = P_{\text{des}}[i]$, while avoiding inter-robot and robot–workspace collisions, i.e., $P(t) \in \mathfrak{C}^f$ for all $t \geq 0$.*

## 3. Control Design

To address the aforementioned problem, first, we employ a hierarchical cell decomposition scheme for partitioning the configuration space of the multi-robot system $\mathfrak{C}$ into cells, as described in Section 3.1. Then, we design a high-level planner, in Section 3.2, which recursively expands the aforementioned structure until a sequence of adjacent cells connecting $P_{\text{init}}$ and $P_{\text{des}}$ is found. Finally, the low-level control scheme that ensures safe transition between cells until the goal configuration is reached is presented in Section 3.3.

### *3.1. Configuration Space Decomposition*

In this subsection, we present the hierarchical cell decomposition scheme that will be employed in our approach. We begin with disregarding inter-robot collisions and considering the configuration space of each individual robot. Particularly, the configuration space of robot $i$, denoted herein by $\mathcal{A}_i(\mathcal{W})$, corresponds to the largest subset of $\mathcal{W}$ where the reference point of robot $i$ can be placed such that $\mathcal{R}_i(p_i) \cap \mathcal{W}^o = \emptyset$, for all $p_i \in \mathcal{A}_i(\mathcal{W})$. Moreover, given a subset $\mathcal{Z}$ of $\mathcal{W}$, we shall use $\mathcal{A}_i(\mathcal{Z})$ to denote the set of feasible positions of robot $i$ which belong to $\mathcal{Z}$, i.e.,

$$\mathcal{A}_i(\mathcal{Z}) \triangleq \{\, p \mid p \in \mathcal{Z} \text{ and } \mathcal{R}_i(p) \cap \mathcal{W}^o = \emptyset \,\},\ \forall i \in \mathfrak{I}_{N_\mathcal{R}}. \tag{3}$$

In addition to $\mathcal{A}_i(\cdot)$, which corresponds to the set of feasible positions of robot $i$, we also consider two estimations of the area that is potentially occupied by $\mathcal{R}_i$ when $p_i$ is restricted in a subset $\mathcal{Z}$ of $\mathcal{W}$. Particularly, given a robot $i \in \mathfrak{I}_{N_\mathcal{R}}$ and a set $\mathcal{Z} \subseteq \mathcal{A}_i(\mathcal{W})$, let $\overline{\mathcal{R}_i}(\mathcal{Z})$ and $\underline{\mathcal{R}_i}(\mathcal{Z})$ be an over-approximation and under-approximation, respectively, of the footprint of $\mathcal{R}_i$ when robot $i$ is swept over $\mathcal{Z}$ such that:

$$\begin{aligned} \overline{\mathcal{R}_i}(\mathcal{Z}) &\supseteq \bigcup_{p \in \mathcal{Z}} \mathcal{R}_i(p) \\ \underline{\mathcal{R}_i}(\mathcal{Z}) &\subseteq \bigcap_{p \in \mathcal{Z}} \mathcal{R}_i(p) \end{aligned} \tag{4}$$

and

$$\begin{aligned} \overline{\mathcal{R}_i}(\mathcal{Z}) &\subseteq \overline{\mathcal{R}_i}(\mathcal{Z}'), \quad \forall \mathcal{Z} \subseteq \mathcal{Z}' \\ \underline{\mathcal{R}_i}(\mathcal{Z}) &\supseteq \underline{\mathcal{R}_i}(\mathcal{Z}'), \quad \forall \mathcal{Z} \subseteq \mathcal{Z}'. \end{aligned} \tag{5}$$

An example of such approximations can be seen in Figure 2. (for disk-shaped robots, a valid over-approximation $\overline{\mathcal{R}_i}(\mathcal{Z})$ can be computed by offsetting $\mathcal{Z}$ by $r_i$, whereas $\underline{\mathcal{R}_i}(\mathcal{Z})$ can be calculated by $\cap_{p \in \partial\mathcal{Z}} \mathcal{R}_i(p)$).

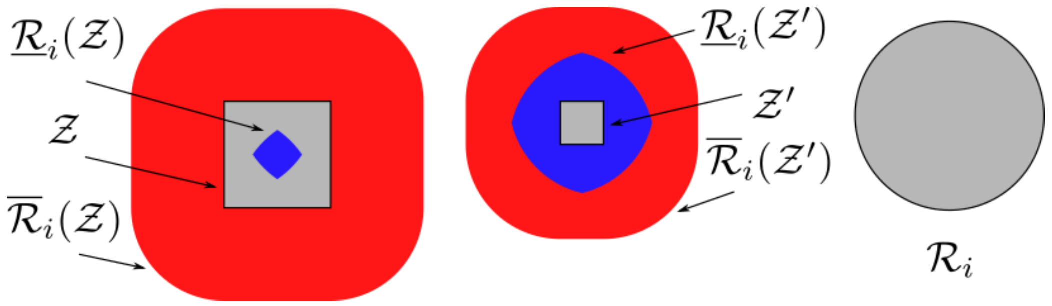

$\underline{\mathcal{R}_i}(\mathcal{Z})$
$\mathcal{Z}$
$\overline{\mathcal{R}_i}(\mathcal{Z})$
$\underline{\mathcal{R}_i}(\mathcal{Z}')$
$\mathcal{Z}'$
$\overline{\mathcal{R}_i}(\mathcal{Z}')$
$\mathcal{R}_i$

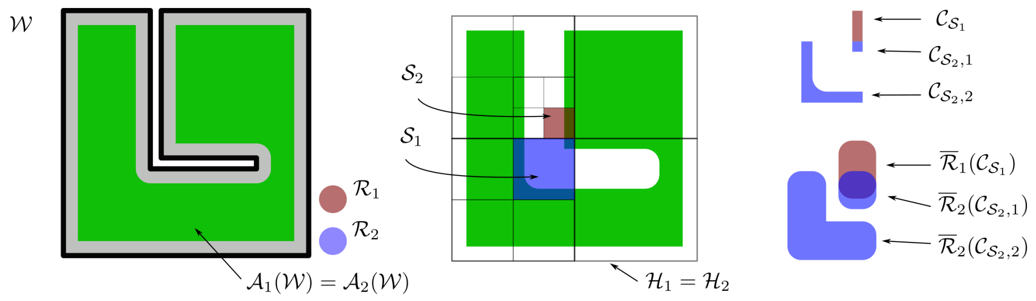
**Figure 2.** Over-approximation $\mathcal{R}_i(\mathcal{Z})$ (resp., $\mathcal{R}_i(\mathcal{Z}')$) and under-approximation $\mathcal{R}_i(\mathcal{Z})$ (resp., $\mathcal{R}_i(\mathcal{Z}')$) of the footprint of robot $i$ when swept over $\mathcal{Z} \subset \mathbb{R}^2$ (resp., $\mathcal{Z}$).

We now consider a set $\mathcal{S} \subseteq \mathbb{R}^2$ that has the form $[x_1, x_2] \times [y_1, y_2]$. We shall refer to such a set as a simple slice of $\mathbb{R}^2$. Given a simple slice $\mathcal{S}$ and a robot $i \in \mathfrak{I}_{N_\mathcal{R}}$, we will use

$\mathcal{W}^i_{\mathcal{S}} \triangleq \mathcal{A}_i(\mathcal{W}) \cap \mathcal{S}$ to denote the set of feasible positions of robot $i$ (neglecting inter-robot collisions) that are contained in $\mathcal{S}$. A set $\mathfrak{S} = \{\, \mathcal{S}_i \mid i \in \mathfrak{I}_{N_{\mathcal{S}}} \,\}$ of $N_{\mathcal{S}}$ simple slices shall be called a cover of $\mathcal{W}$ iff

$$\mathcal{W} = \bigcup_{j \in \mathfrak{I}_{N_{\mathcal{S}}}} \mathcal{S}_j \cap \mathcal{W}. \tag{6}$$

We note that a cover $\mathfrak{S}$ partitions $\mathcal{A}_i(\mathcal{W})$ into a set of regions $\mathcal{W}^i_{\mathcal{S}}$, $\mathcal{S} \in \mathfrak{S}$ each of which consists of zero or more individually connected but pairwise disjoint subsets $\mathcal{C}^i_{\mathcal{S},j}$, $j \in \mathfrak{I}_{N_{\mathfrak{L}}(\mathcal{W}^i_{\mathcal{S}})}$, which shall be referred to as workspace (or simple) cells. A cover $\widehat{\mathfrak{S}} = \{\, (i, \mathfrak{S}_i) \mid i \in \mathfrak{I}_{N_{\mathcal{R}}} \,\}$ of the configuration space $\mathfrak{C}$ is, respectively, defined by assigning a cover to each robot. Accordingly, a configuration space (or compound) slice $\widehat{\mathcal{S}}$ is defined as $\widehat{\mathcal{S}} = \{\, (i, \mathcal{S}_i) \mid i \in \mathfrak{I}_{N_{\mathcal{R}}} \,\}$, where $\mathcal{S}_i$, $i \in \mathfrak{I}_{N_{\mathcal{R}}}$ is a set of simple slices. Likewise, a configuration space cover $\widehat{\mathfrak{S}} = \{\, (i, \mathfrak{S}_i) \mid i \in \mathfrak{I}_{N_{\mathcal{R}}} \,\}$ induces a partitioning of $\mathfrak{C}$ into regions $\widehat{\mathcal{W}}_{\widehat{\mathcal{S}}} \triangleq \mathcal{W}^1_{\mathcal{S}_1} \times \mathcal{W}^2_{\mathcal{S}_2} \times \ldots \times \mathcal{W}^{N_{\mathcal{R}}}_{\mathcal{S}_{N_{\mathcal{R}}}}$, where $\widehat{\mathcal{S}} = \{\, (i, \mathcal{S}_i) \mid i \in \mathfrak{I}_{N_{\mathcal{R}}} \,\}$ is an element of $\widehat{\mathfrak{S}}$. We note that each of these regions may consist of zero or more individually connected but pairwise disjoint subsets $\widehat{\mathcal{C}}_{\widehat{\mathcal{S}},i}$, $i \in \mathfrak{I}_{N_{\mathfrak{L}}(\mathfrak{C}_{\widehat{\mathcal{S}}})}$, which shall be referred to herein as configuration space (or compound) cells. Given the compound cell $\widehat{\mathcal{C}}_{\widehat{\mathcal{S}},i}$, we will use $\widehat{\mathcal{C}}^{[j]}_{\widehat{\mathcal{S}},i}$ to denote its $j$-th component, i.e., $\widehat{\mathcal{C}}^{[j]}_{\widehat{\mathcal{S}},i} \triangleq \mathcal{C}^j_{\mathcal{S}_j,i}$, for all $j \in \mathfrak{I}_{N_{\mathcal{R}}}$. We remark that, unlike hierarchical decomposition schemes commonly encountered in the literature, which use cells of simple geometries (e.g., hypercubes or hyperrectangles), the configuration space cells considered in this work have, in general, arbitrary geometries because their components do not possess a pre-specified shape (see Figure 3). Although this choice renders navigation within a cell $\widehat{\mathcal{C}}$ more complicated, it generally results in coarser partitions because because each component $\widehat{\mathcal{C}}^{[i]}$ of $\widehat{\mathcal{C}}$ belongs in $\mathcal{A}_i(\mathcal{W})$ by construction; thus, the subdivision scheme has to accommodate only for potential inter-robot collisions.



$\mathcal{W}$
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{A}_1(\mathcal{W}) = \mathcal{A}_2(\mathcal{W})$
$\mathcal{S}_2$
$\mathcal{S}_1$
$\mathcal{H}_1 = \mathcal{H}_2$
$\mathcal{C}_{\mathcal{S}_1}$
$\mathcal{C}_{\mathcal{S}_2,1}$
$\mathcal{C}_{\mathcal{S}_2,2}$
$\overline{\mathcal{R}}_1(\mathcal{C}_{\mathcal{S}_1})$
$\overline{\mathcal{R}}_2(\mathcal{C}_{\mathcal{S}_2,1})$
$\overline{\mathcal{R}}_2(\mathcal{C}_{\mathcal{S}_2,2})$

**Figure 3.** Example of a hierarchical configuration space decomposition for a system of two identical robots. The green area corresponds to the configuration space $\mathcal{A}_i(\mathcal{W})$ of each robot. For the sake of simplicity, we assume $\mathcal{H}_1 = \mathcal{H}_2$. The workspace slice $\mathcal{S}_1$ corresponding to robot 1 consists of a single simple cell ($\mathcal{C}_{\mathcal{S}_1}$) whereas slice $\mathcal{S}_2$ corresponding to robot 2 consists of two cells ($\mathcal{C}_{\mathcal{S}_2,1}$, $\mathcal{C}_{\mathcal{S}_2,2}$). The compound cell ($\mathcal{C}_{\mathcal{S}_1}$, $\mathcal{C}_{\mathcal{S}_2,1}$) is labeled as mixed because $\mathcal{R}_1$ and $\mathcal{R}_2$ may intersect when $p_1 \in \mathcal{C}_{\mathcal{S}_1}$ and $p_2 \in \mathcal{C}_{\mathcal{S}_2,2}$ because $\overline{\mathcal{R}_1}(\mathcal{C}_{\mathcal{S}_1}) \cap \overline{\mathcal{R}_2}(\mathcal{C}_{\mathcal{S}_2,1}) \neq \emptyset$, whereas ($\mathcal{C}_{\mathcal{S}_1}$, $\mathcal{C}_{\mathcal{S}_2,2}$) is marked as admissible.

Regarding now the transition between configuration space cells, we introduce some required notions of connectedness. We begin with considering two distinct simple slices $\mathcal{S}_i$ and $\mathcal{S}_j$ which shall be called adjacent iff their intersection $\mathcal{S}_i \cap \mathcal{S}_j$ is not empty. Moreover, let $\mathcal{C}_{\mathcal{S}_m,i}$ and $\mathcal{C}_{\mathcal{S}_n,j}$ be two distinct workspace cells. We define these simple cells as adjacent iff $\mathcal{C}_{\mathcal{S}_m,i} \cap \mathcal{C}_{\mathcal{S}_n,j} \neq \emptyset$. Apparently, $\mathcal{C}_{\mathcal{S}_m,i}$ and $\mathcal{C}_{\mathcal{S}_n,j}$ being adjacent implies that $\mathcal{S}_m$ and $\mathcal{S}_n$ are also adjacent. The aforementioned definitions can be naturally extended to compound slices and cells, as well. Particularly, two compound slices $\widehat{\mathcal{S}}_m = \{\, (i, \mathcal{S}_{m,i}) \mid i \in \mathfrak{I}_{N_{\mathcal{R}}} \,\}$ and $\widehat{\mathcal{S}}_n = \{\, (i, \mathcal{S}_{n,i}) \mid i \in \mathfrak{I}_{N_{\mathcal{R}}} \,\}$ are adjacent iff $\mathcal{S}_{m,i}$, $\mathcal{S}_{n,i}$ are adjacent, for all $i \in \mathfrak{I}_{N_{\mathcal{R}}}$, whereas two compound cells $\widehat{\mathcal{C}}_i$ and $\widehat{\mathcal{C}}_j$ are adjacent iff $\widehat{\mathcal{C}}^{[k]}_i$ and $\widehat{\mathcal{C}}^{[k]}_j$ are adjacent, for all

$k \in \mathfrak{I}_{N_\mathcal{R}}$. A path $\Pi$ of configuration space cells is defined as any finite string of sequentially adjacent compound cells. Obviously, a path $\Pi$ consisting of cells that lie entirely in $\mathfrak{C}^f$ and contain both $P_{\text{init}}$ and $P_{\text{des}}$ is a valid solution to our path finding sub-problem. In order to discover such a path, we build a hierarchical decomposition $\mathfrak{H} = \{ (i, \mathcal{H}) \mid i \in \mathfrak{I}_{N_\mathcal{R}} \}$ of the configuration space $\mathfrak{C}$ by assigning to each robot $i$ a hierarchical partitioning of the workspace $\mathcal{W}$, represented as a connected, directed tree $\mathcal{H} \triangleq (\mathcal{N}_\mathcal{H}, \mathcal{E}_\mathcal{H})$ such that:

- Each node $\mathcal{S} \in \mathcal{N}_\mathcal{H}$ is a simple slice.
- Every child $\mathcal{S}_j$ of a given node $\mathcal{S}_i$ (i.e., $\left(\mathcal{S}_i, \mathcal{S}_j\right) \in \mathcal{E}_\mathcal{H}$) is a strict subset of $\mathcal{S}_i$.
- The set of leaf nodes must form a cover of $\mathcal{W}$.

Finally, an algorithm was devised for appropriately expanding $\mathfrak{H}$ until a solution is found, as described in the following subsection.

### 3.2. High-Level Planner

In this subsection, we present a high-level planner for finding a sequence $\Pi$ of adjacent cells in $\mathfrak{C}^f$ connecting the initial $P_{\text{init}}$ and goal $P_{\text{des}}$ configurations. One of the main advantages of the proposed algorithm is the use of a suitable labeling scheme, which allows it to recursively subdivide, at each iteration, configuration space cells that lie on the boundary between $\mathfrak{C}^f$ and $\mathfrak{C}^o$ while ignoring cells that lie completely inside $\mathfrak{C}^f$ or $\mathfrak{C}^o$. To do so, this labeling scheme exploits the over- and under-approximations $\overline{\mathcal{R}_i}$ and $\underline{\mathcal{R}_i}$ of each robot's footprint, defined in Section 3.1, to determine whether a robot may collide with another one while each robot navigates independently within its respective workspace cell. More specifically, given a compound cell $\widehat{\mathcal{C}}$, the employed cell labeling scheme works as follows:

- If the intersection of all $\overline{\mathcal{R}_i}\left(\widehat{\mathcal{C}}^{[i]}\right)$, $i \in \mathfrak{I}_{N_\mathcal{R}}$ is empty, then, by virtue of (4), no robot may come across another while $P \in \widehat{\mathcal{C}}$; thus, $\widehat{\mathcal{C}}$ is entirely contained in $\mathfrak{C}^f$. Such a compound cell is marked as *admissible*.
- If the intersection of all $\underline{\mathcal{R}_i}\left(\widehat{\mathcal{C}}^{[i]}\right)$, $i \in \mathfrak{I}_{N_\mathcal{R}}$ is non-empty, then, by virtue of (4), for every $P \in \widehat{\mathcal{C}}$ there exists at least one pair of intersecting robots; thus, $\widehat{\mathcal{C}}$ is entirely contained in $\mathfrak{C}^o$. Such a compound cell is marked as *inadmissible*.
- If $\widehat{\mathcal{C}}$ is neither admissible nor inadmissible, it is marked as *mixed*.

In general, mixed cells encapsulate both feasible and infeasible configurations and expanding them (recursively) should yield admissible and inadmissible subcells. On the other hand, by virtue of (5), the subdivision of admissible (resp., inadmissible) cells yields only admissible (resp., inadmissible) cells, without contributing any further in the configuration space's exploration.

The planner's main search algorithm is described in Algorithm 1, which initially constructs a coarse compound slice hierarchical partitioning made of each robot's feasible set $\mathcal{A}_i(\mathcal{W})$, thus enclosing all $\mathfrak{C}^f$ (functions INITIALIZEHIERARCHY and INITIALIZECCELLS). Then, the initially computed compound cells get expanded until admissible ones, containing $P_{\text{init}}$ and $P_{\text{des}}$, are found (function FINDENCLOSINGACCELL), whereas the inability to find such compound cells indicates infeasibility of the given problem and the algorithm terminates. Next, an initial path $\Pi$ connecting $\widehat{\mathcal{C}}_{\text{init}}$ or $\widehat{\mathcal{C}}_{\text{goal}}$ made of compound cells belonging to the exploration's frontier set $S_{\widehat{\mathcal{C}},F}$ (i.e., the set of unexpanded admissible and mixed cells) is built (function CONNECTSTRINGS). At each iteration, the first mixed compound cell of $\Pi$ (function GETFIRSTMIXEDCCELL) is removed from the frontier and is expanded (function EXPANDCCELL) by subdividing the widest simple cell $\mathcal{C}$ whose over-approximation $\overline{\mathcal{R}_i}(\mathcal{C})$ intersects with another (functions GETCONFLICTINGSCELLS and SELECTSCELLWITHWIDESTSSLICE) into smaller ones, as seen in Algorithm 2. Finally, a new path is constructed using standard back-tracking techniques until either $\Pi$ consists only of admissible cells or no new path of mixed and admissible cells leading to $P_{\text{des}}$ can be found.

**Algorithm 1** Planner's Main Algorithm

```
function FINDAPATH( $P_{\text{init}}$, $P_{\text{des}}$ )
    $\mathfrak{H} \leftarrow$ INITIALIZEHIERARCHY
    $S_{\widehat{\mathcal{C}}} \leftarrow$ INITIALIZECCELLS($\mathfrak{H}$)
    $S_{\widehat{\mathcal{C}},F} \leftarrow S_{\widehat{\mathcal{C}}}$
    $\widehat{\mathcal{C}}_{\text{init}}, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F} \leftarrow$ FINDENCLOSINGACCELL( $P_{\text{init}}, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F}$)
    $\widehat{\mathcal{C}}_{\text{goal}}, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F} \leftarrow$ FINDENCLOSINGACCELL( $P_{\text{des}}, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F}$)
    if $\widehat{\mathcal{C}}_{\text{init}}$ is null or $\widehat{\mathcal{C}}_{\text{goal}}$ is null then
        return null
    end if
    $\Pi \leftarrow$ CONNECTSTRINGS( $[\widehat{\mathcal{C}}_{\text{init}}]$, $[\widehat{\mathcal{C}}_{\text{goal}}]$, $S_{\widehat{\mathcal{C}},F}$ )
    while not (($\Pi$ is null) or ISADMISSIBLEP($\Pi$)) do
        $\Pi, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F} \leftarrow$ EXPANDPATH( $\Pi, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F}$ )
    end while
    return $\Pi$
end function
function FINDENCLOSINGACCELL( $P, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F}$ )
Given a point $P$ in the configuration space of the robotic system, this function identifies the cell in the hierarchy $\mathfrak{H}$ that contains that point and, if $S_{\widehat{\mathcal{C}}}$ is mixed, it iteratively subdivides $S_{\widehat{\mathcal{C}}}$ (modifying the hierarchy $\mathfrak{H}$ in the process) until it obtains a subcell $S_{\widehat{\mathcal{C}},F}$ of $S_{\widehat{\mathcal{C}}}$ that contains $P$ and is admissible.
    $\widehat{\mathcal{C}} \leftarrow$ FINDCCELLCONTAININGPOS($P$, $S_{\widehat{\mathcal{C}},F}$)
    while $\widehat{\mathcal{C}}$ is not admissible) do
        $i, \mathcal{S}, \mathcal{C} \leftarrow$ SELECTSCELLWITHWIDESTSSLICE($\widehat{\mathcal{C}}$)
        $S_{\widehat{\mathcal{C}}_R}, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F} \leftarrow$
          EXPANDCCELL( $\widehat{\mathcal{C}}, i, \mathcal{S}, \mathcal{C}, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F}$ )
        $\widehat{\mathcal{C}} \leftarrow$ FINDCCELLCONTAININGPOS($P$, $S_{\widehat{\mathcal{C}},F}$)
    end while
    return $\widehat{\mathcal{C}}, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F}$
end function
function CONNECTSTRINGS( $[\widehat{\mathcal{C}}_{\text{init}}]$, $[\widehat{\mathcal{C}}_{\text{goal}}]$, $S_{\widehat{\mathcal{C}},F}$ )
This function corresponds to some standard graph search algorithm (e.g., depth-first, breadth-first, A*) which, given: a) the initial $[\widehat{\mathcal{C}}_{\text{init}}]$ and desired $[\widehat{\mathcal{C}}_{\text{goal}}]$ compound cells (in case two paths of cells are passed as arguments, the last cell of the first path and the first cell of the second path are used as $[\widehat{\mathcal{C}}_{\text{init}}]$ and $[\widehat{\mathcal{C}}_{\text{goal}}]$, respectively), b) a set of compound cells (nodes) $S_{\widehat{\mathcal{C}},F}$ and their adjacencies (edges), yields either a sequence of admissible and mixed cells connecting $[\widehat{\mathcal{C}}_{\text{init}}]$ and $[\widehat{\mathcal{C}}_{\text{goal}}]$ if such a path exists or null otherwise.
end function
```

### *3.3. Velocity Control Law*

Given now a path $\Pi$ consisting of $N_\Pi$ admissible configuration space cells, we present a distributed control law for safely navigating from one cell to the next until the goal configuration $P_{\text{des}}$ is reached. First, we consider two consecutive compound cells $\widehat{\mathcal{C}}_\ell$ and $\widehat{\mathcal{C}}_{\ell+1}$ in $\Pi$, for which we compute the goal set $\mathcal{G}_{i,\ell} \triangleq \widehat{\mathcal{C}}_\ell^{[i]} \cap \widehat{\mathcal{C}}_{\ell+1}^{[i]}$ of each robot $i$, which contains feasible configurations in both $\widehat{\mathcal{C}}_\ell^{[i]}$ and $\widehat{\mathcal{C}}_{\ell+1}^{[i]}$ and is non-empty by construction. Respectively, the goal set corresponding to the last cell of $\Pi$ consists of just the desired configuration $P_{\text{des}}$, i.e., $\widehat{\mathcal{C}}_{N_\Pi} = \{P_{\text{des}}\}$ (see Figure 4). Furthermore, let $\mathcal{F}_{\ell,i} \triangleq \widehat{\mathcal{C}}_\ell^{[i]} \setminus \operatorname{int}(\mathcal{G}_{\ell,i})$ and $\mathcal{G}'_{\ell,i} \triangleq \mathcal{G}_{\ell,i} \cap \mathcal{F}_{\ell,i}$, for all $k \in \mathfrak{I}_{N_\Pi}$. Notice that $\mathcal{G}'_{\ell,i}$ is generally made of one or more pairwise disjoint subsets of arbitrary connectedness as well as that robot $i$ should navigate to any of these regions without escaping $\widehat{\mathcal{C}}_\ell^{[i]}$ in order to successfully traverse to the next specified workspace cell. When more than one of such disjoint goal subsets are reachable from the connected component of $\mathcal{F}_{\ell,i}$ that contains $p_i$, one of them is arbitrarily (though, deterministically) selected and assigned as the goal set; a more sophisticated approach for choosing goal regions could be employed, but it exceeds the scope of the current work. Respectively, the transition from $\widehat{\mathcal{C}}_\ell$ to $\widehat{\mathcal{C}}_{\ell+1}$ is considered complete after every robot $i$ reaches $\widehat{\mathcal{C}}^{[i]}_{\ell+1}$. We also remark that when $\widehat{\mathcal{C}}^{[i]}_{\ell} \subseteq \widehat{\mathcal{C}}^{[i]}_{\ell+1}$, robot $i$ simply needs to retain its current position during step $\ell$.

**Algorithm 2** Path Expansion at Mixed Compound Cell

```
function ExpandPath( $\Pi$, $\mathfrak{H}$, $S_{\widehat{\mathcal{C}}}$, $S_{\widehat{\mathcal{C}},F}$ )
    $\widehat{\mathcal{C}} \leftarrow$ GetFirstMixedCCell($\Pi$)
    $\mathcal{L}_{\text{pre}}, \mathcal{L}_{\text{suf}} \leftarrow$ SplitString($\Pi, \widehat{\mathcal{C}}$)
    $S_{\mathcal{C}} \leftarrow$ GetConflictingSCells($\widehat{\mathcal{C}}$)
    $i, \mathcal{S}, \mathcal{C} \leftarrow$ SelectSCellWithWidestSSlice($S_{\mathcal{C}}$)
    $S_{\widehat{\mathcal{C}}_R}, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F} \leftarrow$ ExpandCCell( $\widehat{\mathcal{C}}, i, \mathcal{S}, \mathcal{C}, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F}$ )
    $\Pi \leftarrow$ ConnectStrings($\mathcal{L}_{\text{pref}}, \mathcal{L}_{\text{suf}}, S_{\widehat{\mathcal{C}},F}$)
    return $\Pi, \mathfrak{H}, S_{\widehat{\mathcal{C}}}, S_{\widehat{\mathcal{C}},F}$
end function
function SplitString($\Pi, \widehat{\mathcal{C}}$)
Given a path $\Pi$ and a specified cell $\widehat{\mathcal{C}}$ in $\Pi$, return two subsequences $\mathcal{L}_{\text{pre}}$ and $\mathcal{L}_{\text{suf}}$ of $\Pi$
such that $\Pi = (\mathcal{L}_{\text{pre}}, \widehat{\mathcal{C}}, \mathcal{L}_{\text{suf}})$.
end function
function SelectSCellWithWidestSSlice($S_{\mathcal{C}}$)
Given a set of simple cells $S_{\mathcal{C}}$, return the simple cell in $S_{\mathcal{C}}$, the bounding box of which
has the largest length or width.
end function
```

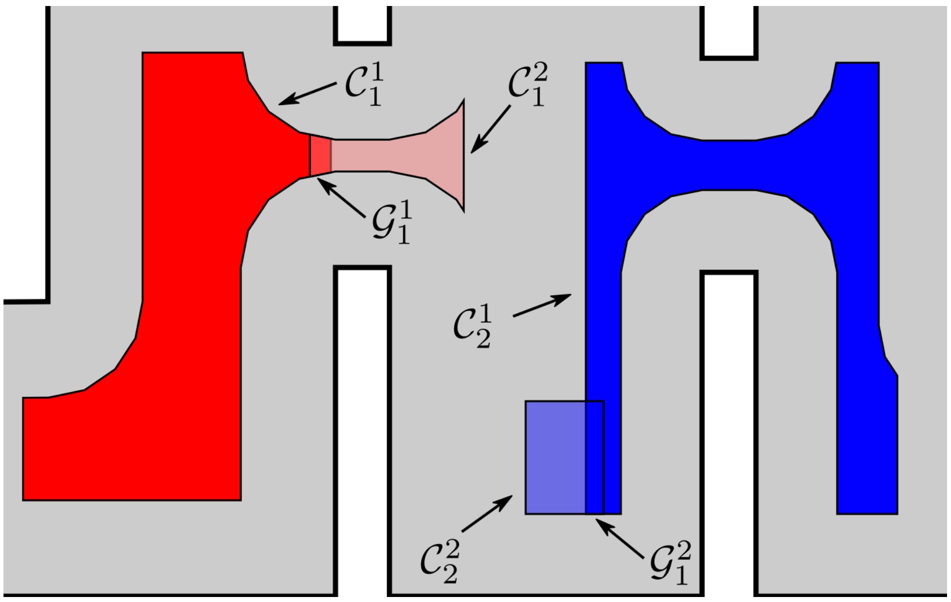


**Figure 4.** Two adjacent compound cells $\widehat{\mathcal{C}}_1 = (\mathcal{C}_1^1, \mathcal{C}_1^2)$ and $\widehat{\mathcal{C}}_2 = (\mathcal{C}_2^1, \mathcal{C}_2^2)$. In order for robot 1 (resp., robot 2) to successfully move from $\mathcal{C}_1^1$ to $\mathcal{C}_2^1$ (resp., from $\mathcal{C}_1^2$ to $\mathcal{C}_2^2$), it has to reach any point of $\mathcal{G}_{1,1}$ (resp., $\mathcal{G}_{2,1}$).

In order to fulfill the aforementioned specifications, we equip each robot $i$ with a controller $u_i$ based on suitable workspace transformations and adaptive artificial potential fields, which were originally presented in [27] and possess guaranteed domain invariance and almost-global convergence properties. More specifically, we build a diffeomorphic transformation $q_i^\ell = T_i^\ell(p_i)$ that maps $\mathcal{F}_{\ell,i}$ to the unit disk $\mathcal{D}$, the outer boundary of $\mathcal{F}_{\ell,i}$ to the unit circle $\partial\mathcal{D}$ and collapses all inner boundaries to distinct points $q_{i,j}^\ell$, $j \in \Im_{N_i^\ell}$, where $N_i^\ell$ is the genus of $\mathcal{F}_{\ell,i}$. We now distinguish the following two cases of possible goal sets: (a) $\mathcal{G}'_{\ell,i}$ being an inner boundary of $\mathcal{F}_{\ell,i}$, and (b) $\mathcal{G}'_{\ell,i}$ being part of the outer boundary of $\mathcal{F}_{\ell,i}$. Depending on the case, $T_i^\ell$ must be appropriately adapted to simplify the subsequent potential field's design. Particularly, case (a) can be accommodated by modifying $T_i^\ell$ such that $\mathcal{G}_{\ell,i}$ collapses to an inner point $q_{\text{des},i}^\ell$ of $\mathcal{D}$, whereas case (b) is addressed by designing $T_i^\ell$ such that $\mathcal{G}'_{\ell,i}$ collapses to a single point $q_{\text{des},i}^\ell$ on $\partial\mathcal{D}$. Next, we define the harmonic potential field $\phi_i^\ell$ used by robot $i$ during step $\ell$ by placing point harmonic sources upon the corresponding goal configuration $q_{\text{des},i}^\ell$ and the transformed inner obstacles $q_{i,j}^\ell$, given by:

$$\phi_i^\ell = k_{i,d}^\ell \ln\left(\frac{\|q_i^\ell - q_{\text{des},i}^\ell\|}{2}\right) - \sum_{j\in\Im_{N_i^\ell}} k_{i,j}^\ell \ln\left(\frac{\|q_i^\ell - q_{i,j}^\ell\|}{2}\right) \tag{7}$$

where $k_{i,d}^\ell > 0$ and $k_{i,j}^\ell \geq 0$ are adaptively varying parameters. Finally, the control law $u_i^\ell$ of robot $i$ during step $\ell$ is given by

$$u_i^\ell = -Ks(q_i^\ell, k_{v,i}^\ell)\left(J_i^\ell(q_i^\ell)\right)^{-1}\nabla_{q_i^\ell}\psi_i^\ell(q_i^\ell, k_{v,i}^\ell) \tag{8}$$

where $K$ is a positive control gain, $J_i^\ell$ is the Jacobian matrix of $T_i^\ell$, $s$ is a factor ensuring collision avoidance with the outer boundary and $\psi_i^\ell = 1 + \tanh(w\phi_i^\ell)/2$, with $w$ a positive constant.

## 4. Simulation Results

In this section, we present the simulation results demonstrating the efficacy of the proposed methodology. As the contribution of this work is a path-planning algorithm, simulations were deemed sufficient to validate the efficacy of the proposed method. To conduct these simulations, the algorithm described in this work was implemented in C++, using Boost Geometry and CGAL for performing the geometric operations, such as the configuration computation, subdivision of cells, etc. The implementation can be found at the following url: https://github.com/maxchaos/mrnav, accessed on 20 December 2022. As described, the configuration space was partitioned into a tree of cells and a standard graph search algorithm was employed for finding admissible paths of mixed cells connecting the initial and goal configurations. After such a path was obtained, for each compound cell in the sequence, we extracted the simple cells corresponding to each robot, and for each robot, we employed the low-level control scheme to drive it to the border of the next cell or its goal configuration in case the current cell was the last of the path. Particularly, we consider five scenarios where a system consisting of 2, 4, 6, 8 and 10 robots, respectively, initialized within the workspace depicted in Figure 5 is requested to reach a specified final configuration. The time required by the proposed planner, as well as the total amount of compound cells generated during the solution of each case, are shown in Table 1. We remark that the planner expanded the mixed compound cells by subdividing the corresponding conflicting simple slice into four identical overlapping sub-slices. The motion profiles executed by the robots in each corresponding case can be seen in Figures 5–9. Additionally, Figures 10 and 11 depict the initial and goal configurations as well as the computed enclosing cells, respectively, for the eight-robot scenario. As one can verify from the figures, the robots navigate successfully to their individual goals. A video of the aforementioned simulated scenarios can be found at the following url: https://youtu.be/asWnKLNX2Eg, accessed on 20 December 2022.

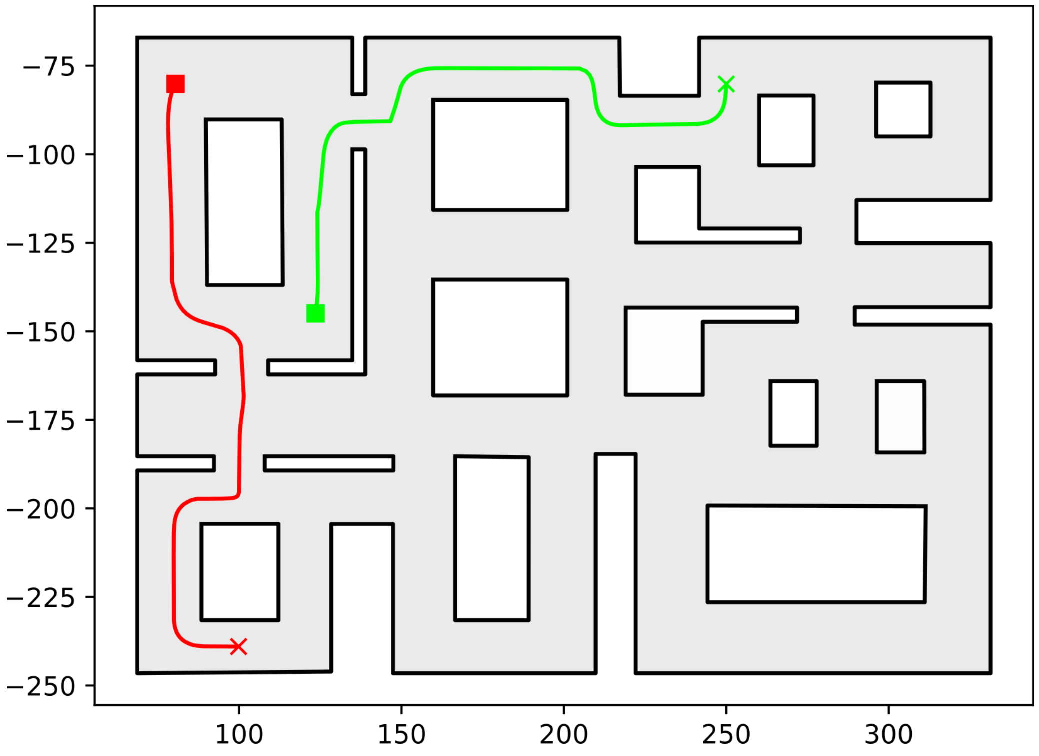


**Figure 5.** Executed trajectories of the two-robot case. Squares indicate initial positions, whereas the corresponding goal positions are depicted using crosses. Robot #1 (red); Robot #2 (green). The axes units are in (dm).

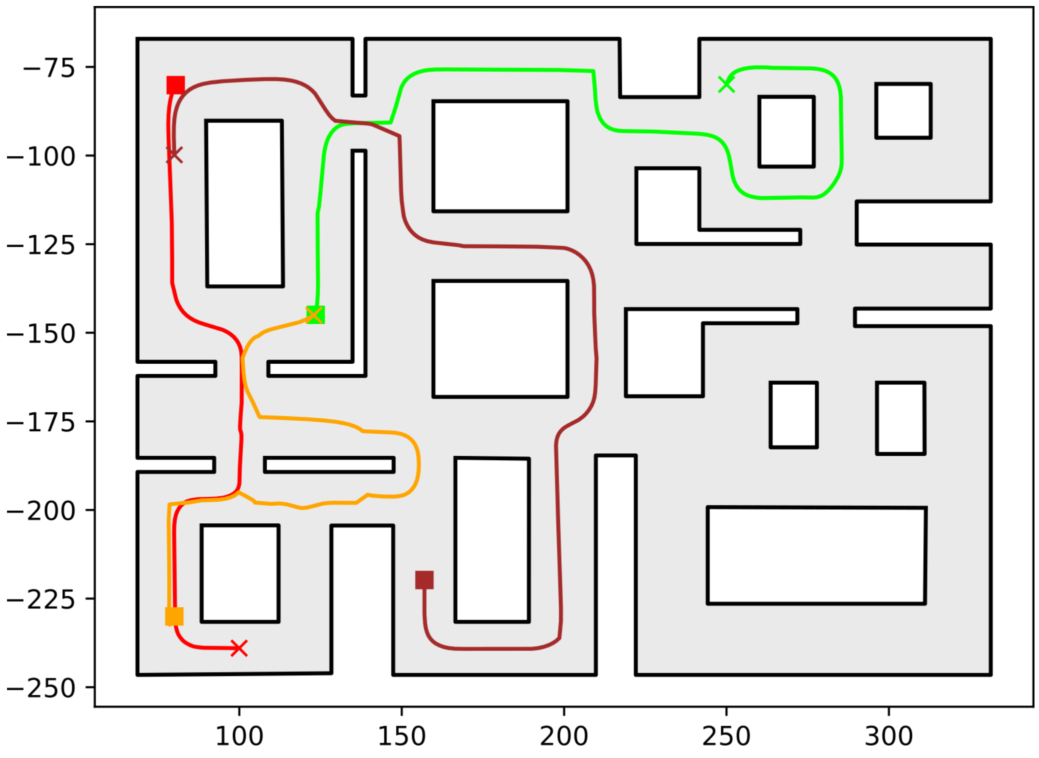


**Figure 6.** Executed trajectories of the four-robot case. Squares indicate initial positions, whereas the corresponding goal positions are depicted using crosses. Robot #1 (red); Robot #2 (green); Robot #3 (dark red); Robot #4 (orange). The axes units are in (dm).

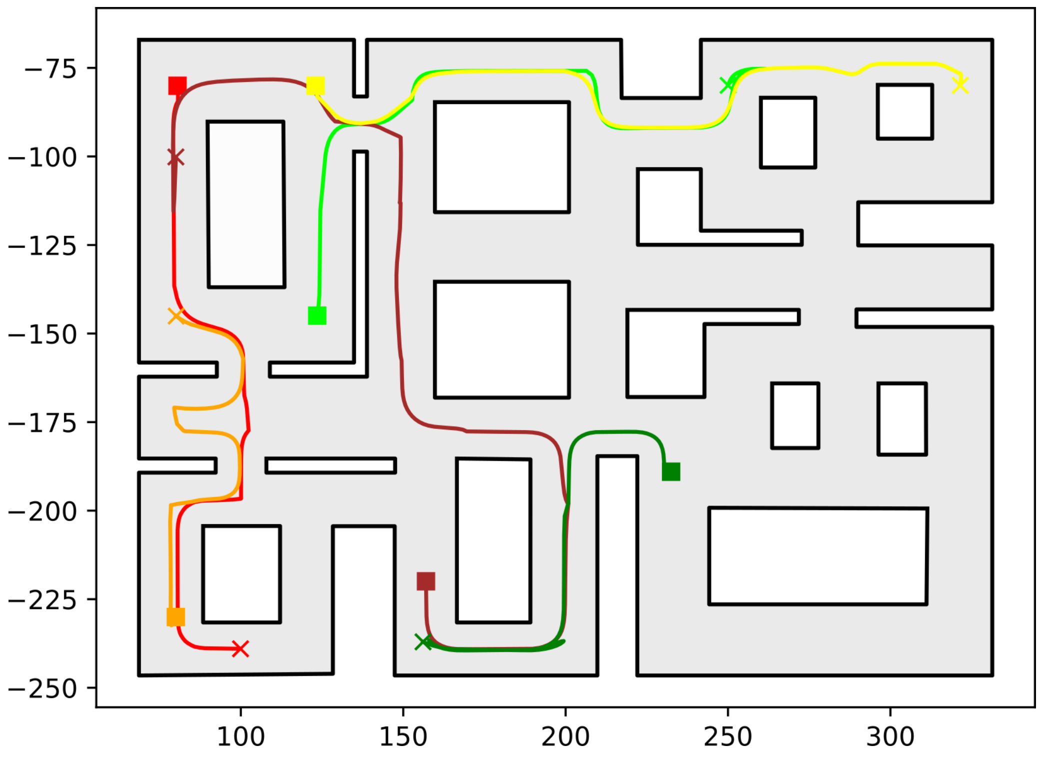


**Figure 7.** Executed trajectories of the six-robot case. Squares indicate initial positions, whereas the corresponding goal positions are depicted using crosses. Robot #1 (red); Robot #2 (green); Robot #3 (dark red); Robot #4 (orange); Robot #5 (yellow); Robot #6 (dark green). The axes units are in (dm).

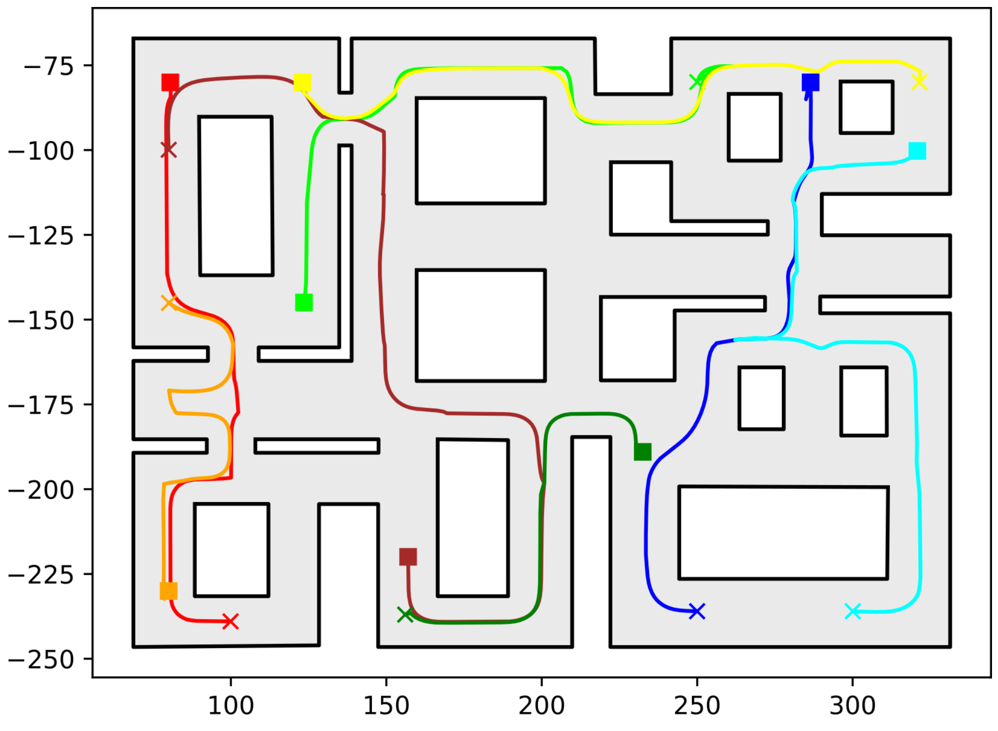


**Figure 8.** Executed trajectories of the eight-robot case. Squares indicate initial positions, whereas the corresponding goal positions are depicted using crosses. Robot #1 (red); Robot #2 (green); Robot #3 (dark red); Robot #4 (orange); Robot #5 (yellow); Robot #6 (dark green); Robot #7 (blue); Robot #8 (cyan). The axes units are in (dm).

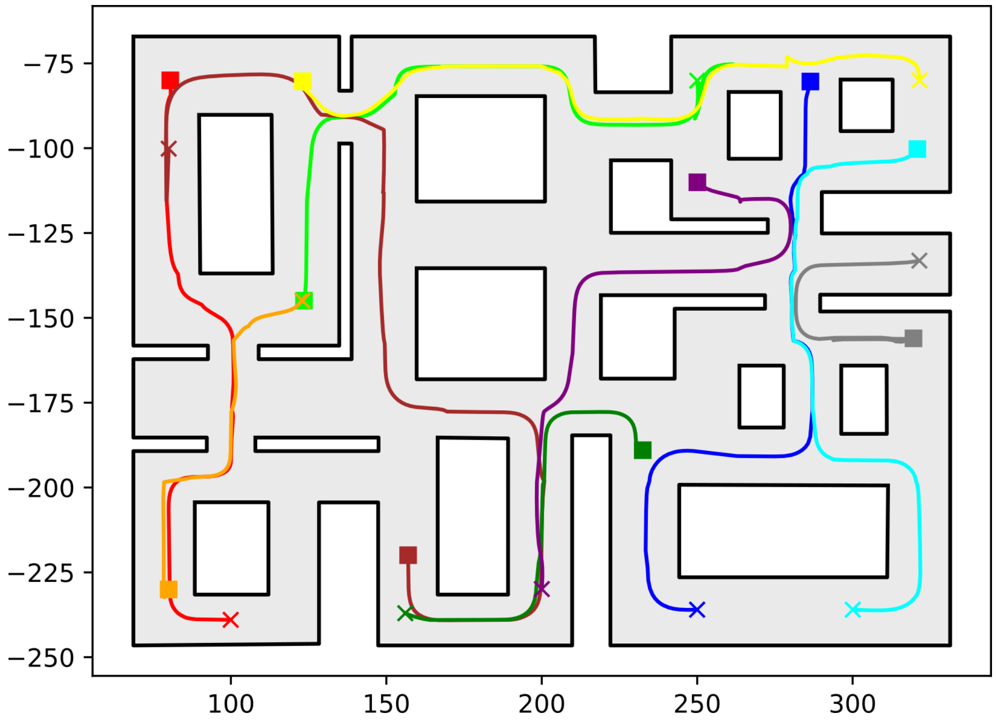


**Figure 9.** Executed trajectories of the ten-robot case. Squares indicate initial positions, whereas the corresponding goal positions are depicted using crosses. Robot #1 (red); Robot #2 (green); Robot #3 (dark red); Robot #4 (orange); Robot #5 (yellow); Robot #6 (dark green); Robot #7 (blue); Robot #8 (cyan); Robot #9 (purple); Robot #10 (grey). The axes units are in (dm).

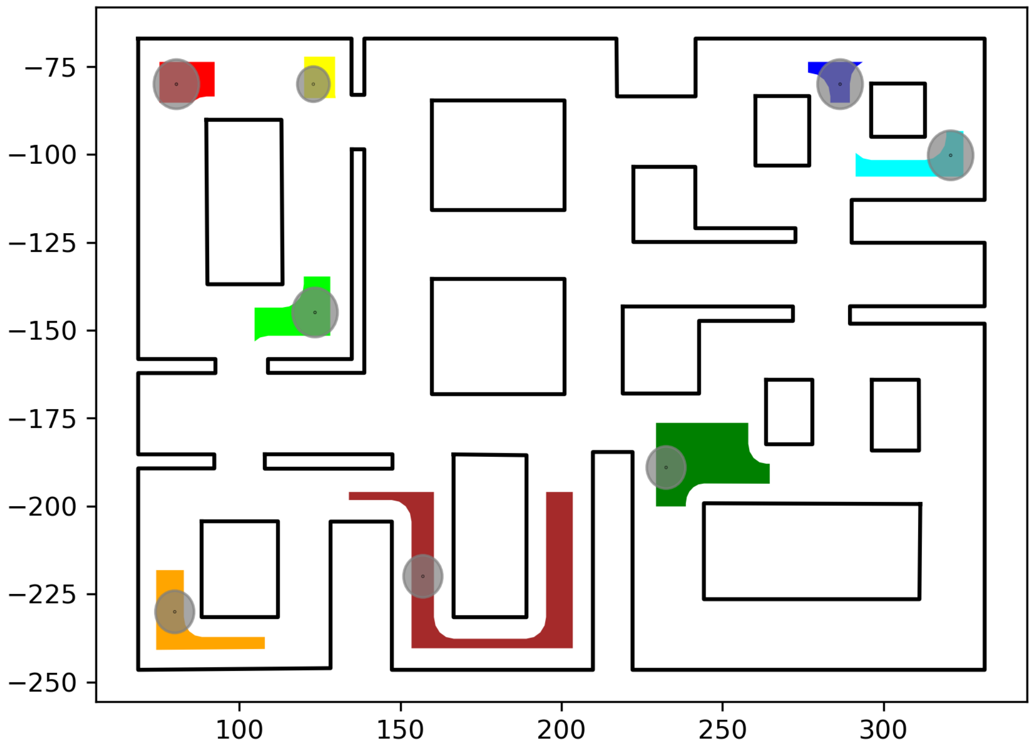


**Figure 10.** The initial robot positions $p_{\text{init},i}$, $i \in \mathfrak{I}_8$ and calculated initial compound cell $\widehat{\mathcal{C}}_{\text{init}}$ . Robot #1 (red); Robot #2 (green); Robot #3 (dark red); Robot #4 (orange); Robot #5 (yellow); Robot #6 (dark green); Robot #7 (blue); Robot #8 (cyan). The axes units are in (dm).

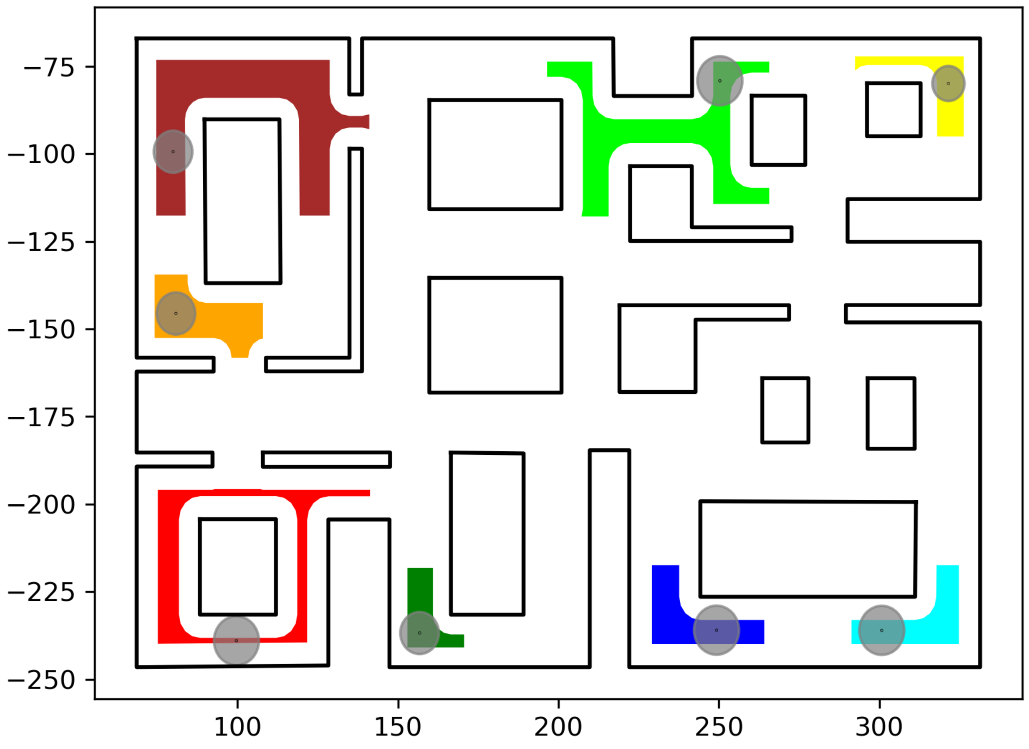


**Figure 11.** The desired robot positions $p_{\text{des},i}$, $i \in \mathfrak{I}_8$ and calculated goal compound cell $\widehat{\mathcal{C}}_{\text{goal}}$. Robot #1 (red); Robot #2 (green); Robot #3 (dark red); Robot #4 (orange); Robot #5 (yellow); Robot #6 (dark green); Robot #7 (blue); Robot #8 (cyan). The axes units are in (dm).

To strengthen our theoretical findings, we also conducted a comparative simulation study over the same scenarios employing a state-of-the-art high-level planner called RRT* [4] that is based on a probabilistic sampling algorithm to extract feasible paths toward the goal configuration. Because RRT* is probabilistic, we ran 50 trials on each scenario and extracted the mean value of the execution time and total path as described in Table 2. We have to stress that owing to the narrow passages of the workspace, as the number of robots increased (particularly for the cases of 8 and 10 robots), the RRT* algorithm reached the maximum terminating condition (an execution time more than 1000 s) very frequently (many runs did not discover feasible paths) without providing feasible solutions. Notice that our algorithm is superior in terms of the execution time and completeness without however sacrificing the performance in terms of the total path length. It should be pointed out that although our algorithm does not incorporate any optimization in either the path calculation or the transition execution, the achieved performance is comparable to the RRT* algorithm, which finds the shortest path among multiple feasible ones.

**Table 1.** Execution time, total length of the robots' paths and number of generated compound cells required by the high-level planner for solving each scenario.

| **Number of Robots** | **2** | **4** | **6** | **8** | **10** |
|---|---|---|---|---|---|
| Time (sec) | 0.088 | 0.240 | 0.845 | 1.36 | 31.1 |
| Length (dm) | 404.94 | 1076.23 | 1396.20 | 1739.53 | 2125.36 |
| Compound Cells | 51 | 348 | 823 | 1014 | 3363 |

**Table 2.** Average and standard deviation for the execution time and total length of the robots' paths over the number of successful runs yielded by the RRT* planner [4] for solving each scenario over 50 runs.

| **Number of Robots** | **2** | **4** | **6** | **8** | **10** |
|---|---|---|---|---|---|
| Time-avg (sec) | 0.041 | 0.124 | 12.375 | 724.136 | 961.230 |
| Time-std (sec) | 0.001 | 0.045 | 0.454 | 30.032 | 41.211 |
| Length-avg (dm) | 380.15 | 1112.21 | 1256.82 | 1712.45 | 1995.63 |
| Length-std (dm) | 1.25 | 5.18 | 7.32 | 15.82 | 17.31 |
| Successful runs | 50 | 50 | 49 | 36 | 5 |

## 5. Conclusions

In this work, we presented a hybrid control scheme for addressing the navigation problem of a team of robots operating within an obstacle-cluttered planar workspace. Particularly, a high-level planner was designed for computing a sequence of feasible cells by adaptively subdividing the system's configuration space using a hierarchical cell decomposition scheme. In addition, a low-level control law based on harmonic potentials and workspace transformations was employed to implement the given plan, safely navigating each robot to its specified goal. The contribution lies on the adaptive subdivision algorithm for obtaining a sequence of traversable cells connecting the given initial and final configurations or determining that no solution exists. Unlike standard grid/cell-based methods, the proposed algorithm adaptively subdivides the cells, requiring no selection of some arbitrary resolution by the user. Moreover, unlike probabilistic sampling methods, the proposed method does not generate paths of configuration but rather sequences of safe sets, allowing the user to make use of a larger set of controllers in addition to standard trajectory tracking schemes. In addition, unlike probabilistic sampling methods which are inherently incapable of determining the infeasibility of a given problem (such algorithms require the user to specify an arbitrary upper bound of samples after which the algorithms abort) as a consequence of their probabilistic completeness nature and the fact that sample points should lie within the configuration space, our algorithm partitions the configuration space into the cells (dense subsets) of the configuration space and is capable of determining when no admissible path exists in finite time. Finally, comparative simulation studies were conducted for various scenarios with a state-of-the-art algorithm that validates the proposed scheme's efficacy and demonstrates its superiority in terms of complexity while yielding a satisfactory performance in terms of path lengths.

Regarding the limitations of the proposed approach and future research directions toward healing them, we aim to improve the proposed planner by accommodating for redundant steps appearing in the computed path and adapt it for use in a more distributed fashion, i.e., resolve the conflicts between antagonistically operating robots as they appear. Additionally, introducing optimality in the selection of the sequence of cells as well as in the execution of the transitions deserves further investigation to improve the efficiency of the proposed method. In the same vein, increasing the dimensionality of the problem (considering 3D workspaces and potentially the orientation) would also favor its applicability. Alternatively, a modern approach such as deep reinforcement learning [28] will be sought to examine new ways of solving the multi-robot coordination problem.

**Author Contributions:** Methodology, P.V. and C.P.B.; Validation, P.V.; Formal analysis, P.V. and C.P.B.; Writing—original draft, P.V.; Writing—review & editing, C.P.B. and K.J.K.; Supervision, C.P.B. and K.J.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** The publication fees of this manuscript have been financed by the Research Council of the University of Patras.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Schwartz, J.T.; Sharir, M. On the Piano Movers' Problem: Iii. Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Moving Amidst Polygonal Barriers. *Int. J. Robot. Res.* **1983**, *2*, 46–75. [CrossRef]
2. Canny, J.F. *The Complexity of Robot Motion Planning*; MIT Press: Cambridge, MA, USA, 1988.
3. Veras, L.G.D.O.; Medeiros, F.L.L.; Guimaraes, L.N.F. Systematic Literature Review of Sampling Process in Rapidly-Exploring Random Trees. *IEEE Access* **2019**, *7*, 50933–50953. [CrossRef]
4. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
5. Arslan, O.; Pacelli, V.; Koditschek, D.E. Sensory steering for sampling-based motion planning. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 3708–3715. [CrossRef]
6. Salzman, O.; Hemmer, M.; Raveh, B.; Halperin, D. Motion Planning Via Manifold Samples. *Algorithmica* **2013**, *67*, 547–565. [CrossRef]
7. Salzman, O.; Hemmer, M.; Halperin, D. On the Power of Manifold Samples in Exploring Configuration Spaces and the Dimensionality of Narrow Passages. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 529–538. [CrossRef]
8. Peasgood, M.; Clark, C.; McPhee, J. A Complete and Scalable Strategy for Coordinating Multiple Robots Within Roadmaps. *IEEE Trans. Robot.* **2008**, *24*, 283–292. [CrossRef]
9. Velagapudi, P.; Sycara, K.; Scerri, P. Decentralized prioritized planning in large multirobot teams. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 4603–4609. [CrossRef]
10. Wiktor, A.; Scobee, D.; Messenger, S.; Clark, C. Decentralized and complete multi-robot motion planning in confined spaces. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 1168–1175. [CrossRef]
11. Cap, M.; Novak, P.; Kleiner, A.; Selecky, M. Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 835–849. [CrossRef]
12. Yu, J.; LaValle, S.M. Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics. *IEEE Trans. Robot.* **2016**, *32*, 1163–1177. [CrossRef]
13. Svestka, P.; Overmars, M.H. Coordinated Path Planning for Multiple Robots. *Robot. Auton. Syst.* **1998**, *23*, 125–152. [CrossRef]
14. Van Den Berg, J.P.; Overmars, M.H. Prioritized motion planning for multiple robots. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Edmonton, AB, Canada, 2–6 August 2005; pp. 430–435.
15. Wagner, G.; Choset, H. M*: A complete multirobot path planning algorithm with performance bounds. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 3260–3267. [CrossRef]
16. Wagner, G.; Kang, M.; Choset, H. Probabilistic path planning for multiple robots with subdimensional expansion. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 2886–2892. [CrossRef]
17. Solovey, K.; Salzman, O.; Halperin, D. Finding a Needle in an Exponential Haystack: Discrete RRT for Exploration of Implicit Roadmaps in Multi-robot Motion Planning. In *Springer Tracts in Advanced Robotics*; Springer International Publishing: London, UK , 2015; pp. 591–607. [CrossRef]
18. Shome, R.; Solovey, K.; Dobson, A.; Halperin, D.; Bekris, K.E. Drrt*: Scalable and Informed Asymptotically-Optimal Multi-Robot Motion Planning. *Auton. Robot.* **2019**, *44*, 443–467. [CrossRef]
19. Brooks, R.A.; Lozano-Pérez, T. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Trans. Syst. Man, Cybern.* **1985**, *SMC-15*, 224–233. [CrossRef]
20. Lozano-Perez, T. Spatial Planning: A Configuration Space Approach. *IEEE Trans. Comput.* **1983**, *C-32*, 108–120. [CrossRef]
21. Lozano-Perez, T. A simple motion-planning algorithm for general robot manipulators. *IEEE J. Robot. Autom.* **1987**, *3*, 224–238. [CrossRef]
22. De Berg, M.; Cheong, O.; Van Kreveld, M.; Overmars, M. *Computational Geometry: Algorithms and Applications*; Springer: New York, NY, USA, 2008; pp. 1–386.
23. Zhu, D.J.; Latombe, J. New heuristic algorithms for efficient hierarchical path planning. *IEEE Trans. Robot. Autom.* **1991**, *7*, 9–20. [CrossRef]
24. Vlantis, P.; Vrohidis, C.; Bechlioulis, C.P.; Kyriakopoulos, K.J. Orientation-Aware Motion Planning in Complex Workspaces using Adaptive Harmonic Potential Fields. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 8592–8598. [CrossRef]
25. Swingler, A.; Ferrari, S. A cell decomposition approach to cooperative path planning and collision avoidance via disjunctive programming. In Proceedings of the 49th IEEE Conference on Decision and Control (CDC), Atlanta, GA, USA, 15–17 December 2010; pp. 6329–6336. [CrossRef]

26. Lin, S.; Liu, A.; Wang, J.; Kong, X. A Review of Path-Planning Approaches for Multiple Mobile Robots. *Machines* **2022**, *10*, 773. [CrossRef]
27. Vlantis, P.; Vrohidis, C.; Bechlioulis, C.P.; Kyriakopoulos, K.J. Robot Navigation in Complex Workspaces Using Harmonic Maps. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 1726–1731. [CrossRef]
28. Lin, G.; Zhu, L.; Li, J.; Zou, X.; Tang, Y. Collision-free path planning for a guava-harvesting robot based on recurrent deep reinforcement learning. *Comput. Electron. Agric.* **2021**, *188*, 106350. [CrossRef]