Robotics and Autonomous Systems 103 (2018) 93–110

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Admissible gap navigation: A new collision avoidance approach

Muhannad Mujahed *, Dirk Fischer, Bärbel Mertsching

GET-Lab, University of Paderborn, Pohlweg 47-49, D-33098 Paderborn, Germany

HIGHLIGHTS

- A new concept for collision avoidance, the Admissible Gap (AG), is introduced.
- The exact robot shape and kinematic constraints are taken into account.
- An efficient and stable methodology for extracting gaps is proposed.
- An outstanding navigation performance in unknown dense environments is achieved.
- The AG approach is evaluated and compared with three state-of-the-art methods.

ARTICLE INFO

Article history:
Received 8 October 2017
Received in revised form 25 December 2017
Accepted 12 February 2018
Available online 27 February 2018

Keywords:
Mobile robot
Collision avoidance
Reactive navigation
Laser rangefinder
Sensor-based motion planning

ABSTRACT

This paper proposes a new concept, the *Admissible Gap* (AG), for reactive collision avoidance. A gap is called admissible if it is possible to find a collision-free motion control that guides a robot through it, while respecting the vehicle constraints. By utilizing this concept, a new navigation approach was developed, achieving an outstanding performance in unknown dense environments. Unlike the widely used gap-based methods, our approach directly accounts for the exact shape and kinematics, rather than finding a direction solution and turning it later into a collision-free admissible motion. The key idea is to analyze the structure of obstacles and virtually locate an admissible gap, once traversed, the robot makes progress towards the goal. For this purpose, we introduce a strategy of traversing gaps that respect the kinematic constraints and provides a compromise between path length and motion safety. We also propose a new methodology for extracting gaps that eliminates useless ones, thus reducing oscillations. Experimental results along with performance evaluation demonstrate the outstanding behavior of the proposed AG approach. Furthermore, a comparison with existing state-of-the-art methods shows that the AG approach achieves the best results in terms of efficiency, robustness, safety, and smoothness.

1. Introduction

Mobile robots have proven themselves tremendously useful in a wide variety of real-world applications, such as transportation, search and rescue, and mining. Perhaps the most interesting aspect of these robots is the ability to execute tasks that are difficult or dangerous to be performed by humans. Designing such robots requires to solve several challenges such as detection, grasping, and control. Nevertheless, whatever the task to be carried out, at some point, the robot has to move. Therefore, autonomous navigation is at the heart of any robotic system and has been thoroughly studied since the beginning of robotics.

The difficulties of autonomous navigation arise from the fact that real-world environments are often unpredictable, unstructured, and changes over time. Moreover, moving obstacles may block the robot's working area while performing tasks. Under these circumstances, it is essential to incorporate the sensory information into the control loop, bridging the gap between path planning and motion execution. By this means, the environmental changes are detected in real-time enabling robots to avoid unforeseen obstacles. These difficulties are tackled by reactive collision avoidance methods.

The majority of collision avoidance techniques present limited capability of driving robots through narrow spaces in cluttered environments. This is due the fact that these methods experience several classical problems such as being prone to local minima, failure of steering a robot among closely spaced obstacles, and the tendency to generate oscillatory motion [1]. It has been shown that using some form of high-level description of the sensory information is a successful approach to deal with these environments. The

* Corresponding author.
E-mail addresses: mujahed@get.uni-paderborn.de (M. Mujahed), fischer@get.uni-paderborn.de (D. Fischer), mertsching@get.uni-paderborn.de (B. Mertsching).

https://doi.org/10.1016/j.robot.2018.02.008
0921-8890/

so-called *gap-based* methods [1–4] follow this strategy. However, these methods provide direction solutions assuming holonomic and disc-shaped robots. This is indeed a strong assumption since ignoring the actual vehicle shape may lead to collisions or failure of finding a direction solution. Additionally, ignoring the kinematic constraints may result in computing infeasible motions, relying on approximations when applied on a real vehicle. Hence, accounting for these constraints is of great importance, particularly for robots that are performing tasks in hazardous environments.

In order to deal with this limitation, some methods turn the holonomic solution into a motion control that complies with the shape and kinematic constraints. For instance, in [5] a least squares method is used to align the direction solution with the robot's heading. In [6], a similar approach is proposed by splitting the problem into subproblems (motion, shape, and kinematics). In fact, these solutions are subject to approximations and deal with each constraint separately [7]. This may arise some problems, especially in scenarios requiring high maneuverability. A more convenient solution is proposed in [7] by mapping the workspace into the so called Arc Reachable Manifold (ARM) in such away that, when the navigation method is applied in ARM, these constraints are implicitly considered. Although this approach is general and can be applied to many existing techniques, it has some shortcomings that might limit its use in dense environments: first, constructing the obstacle region in ARM is based on the assumption that the configurations are attainable by elemental circular paths only. Hence, navigable gaps may appear blocked in ARM.[1] Second, the coordinates of ARM are transformed to comply with the kinematic constraints. Searching for openings in the new coordinates is unnatural and may result in detecting incorrect or phantom gaps.

This paper introduces a new concept, the *Admissible Gap* (AG), for reactive collision avoidance. We call a gap admissible if it is possible to find a single motion control that safely guides a robot through it, while obeying the vehicle constraints. By employing the AG concept, it has been possible to develop a collision avoidance approach that successfully drives a robot in unknown dense environments. As compared with other gap-based methods, our approach directly considers the exact shape and kinematic constraints rather than finding a direction solution and then aligning the vehicle with that direction. The basic idea is to extract the set of gaps surrounding the vehicle and select the most promising one in terms of reaching the goal. A virtual *admissible* gap is then constructed in an iterative manner, such that traversing it leads to the selected gap and a compromise between path length and motion safety is achieved.

The admissible gap has appeared in part in [8]. In this paper, the concept is extended by considering all virtual gaps that are constructed in the iterative process, not only one. Consequently, the smoothness of the trajectories has been improved. Furthermore, we propose a new methodology for detecting gaps that is applicable to limited and full field of view sensor types. With this methodology, the total number of gaps is reduced by eliminating useless ones, increasing the stability of navigation and alleviating the possibility of oscillation. Additionally, this paper includes a detailed presentation of the overall method along with useful remarks which have been omitted in the conference for the sake of brevity. Finally, several experiments in dense environments are provided, where outstanding results have been achieved, outperforming existing state-of-the-art techniques in terms of efficiency, safety, robustness, and smoothness. By employing the AG approach, our team successfully competed in the 2016 World RoboCup Rescue League, where we ranked the 3rd place in our first participation in the competition.

The remainder of the paper is structured as follows. Section 2 presents the related work while Section 3 introduces some preliminary definitions. In Section 4, our methodology of extracting gaps is described, and subsequently, in Section 5 the Admissible Gap concept is presented. Section 6 shows how this concept is used to navigate a mobile robot. In Sections 7 and 8, the experimental results are discussed and the performance of the AG is evaluated. Finally, Section 9 points out some concluding remarks and presents recommendations for future work.

2. Related work

Robot motion planning has been thoroughly studied by the robotics community and has been traditionally addressed from two distinct perspectives; path planning (map-based) and collision avoidance (sensor-based). Since path planning is beyond the scope of this paper, only collision avoidance methods are covered (for those interested, refer to [9]). For brevity, the focus will be restricted to some representative approaches, including those that have proved popular across the years and those that are related to the proposed approach. For a thorough description of other methods, the reader can refer to [10].

Perhaps the majority of collision avoidance methods are based on the Artificial Potential Field (APF) concept [11–14], initially introduced by Khatib [15]. Within this concept, the robot is a particle attracted towards the goal and repelled from obstacles. The motion direction is determined based on the vector sum of the attractive and repulsive forces. APFs, although simple to implement and computationally efficient, are prone to local minima and may lead to an oscillatory motion [16]. Many research efforts were devoted towards solving the local minima problem, such as [17] by performing a random walk mechanism or [18] by using a navigation function. Bounini et al. [19] solved the problem by adding some extra repulsive potential, inspired from pouring a liquid with high pressure. Other researchers addressed the oscillation problem, such as [20] by employing a modification of Newton's method or [21,22] by utilizing a class of vector fields. A recent work has extended the APF capabilities to moving obstacles [23]. Although these methods solve the APF drawbacks, they are either subject to strict assumptions or limited to certain scenarios [24].

The family of Bug algorithms (e.g. [25–27]) are among the earliest and simplest reactive obstacle avoidance methods. The key idea of these algorithms is to drive the robot towards the target unless an obstacle is encountered, in this case the robot moves unidirectionally along the obstacle boundary until motion towards the target is once again allowable. The transition between both motion cases is controlled by a globally convergent criterion. A well-known variant, the tangent Bug [28,29], builds a graph of the robot's surroundings using a ray based sensor model. This helps in minimizing the path length as shortcuts can be made while circumnavigating obstacles. Bug algorithms allow robots to move in previously unknown environments with the guarantee that the goal is reached once it is reachable. However, they assume a point-like robot and strongly depend on the sensor accuracy. Moreover, they have only been tested in static environments which is not the case in real-world scenarios.

Other research efforts were directed towards incorporating the motion constraints into the obstacle avoidance problem, selecting an admissible velocity rather than a steering direction. The outcome of these efforts was the development of various methods [30–32], known later as *velocity space* methods. However, it was the Dynamic Window Approach DWA [33,34], initially proposed in [35], that won popularity among the scientific community. It formulates the obstacle avoidance problem as a constraint

[1] A gap is navigable in ARM only if there is a collision-free circular path that is tangent to the robot's heading and passes through both the gap and the robot's origin. In other words, only those gaps that are directly navigable from the current robot's location can be seen in ARM.

optimization in the velocity space. Thus, it might be described as a planning method with a prediction time horizon of a single step [36]. Extensions to other vehicle models and shapes have been proposed in [37]. Optimal paths have also been achieved by employing a navigation function [38,39]. An interesting recent variant [40] proposed a new optimization methodology that favors trajectories similar to human ones. By using the DWA, robots can navigate at higher speeds and show a smooth trajectory. However, they may fail to travel through narrow spaces. Moreover, the DWA requires parameter tuning and is still prone to local minima.

A wide variety of methods, e.g. [41–43], build on the concept of Velocity Obstacles VO proposed in [44]. A VO is basically the set of vehicle velocities that may result in collision at some future time, once determined, a velocity outside of this set is chosen. A similar concept, the Inevitable Collision States, has been proposed [45,46], avoiding the set of forbidden states rather than velocities and reasoning over an infinite time-horizon. The VO concept has been extended to multi-robot navigation [47,48]. An extension to Dubins-like mobile robots has been proposed in [49]. Other recent VO variants addressed the problems of sensing limitations and uncertainties [50,51]. Although these methods explicitly account for the velocity of obstacles, they assume a complete knowledge of the scene. This is, however, hard to achieve in real-world scenarios [52]. Furthermore, it is essential in VO-based methods to select a suitable time horizon which is difficult, particularly in dense and cluttered environments [53].

One interesting concept which has shown enormous potential in mobile robotic applications is gaps: open areas among obstacles through which the vehicle may pass. The Nearness-Diagram (ND) [1,54] was perhaps the first *gap-based* collision avoidance approach. It employs a predefined set of condition states to describe all possible navigational situations and their corresponding actions. The behavior of ND in open areas has been enhanced in [55]. Some variants [2,56] compute the moving direction based on the configuration of all obstacle points, improving smoothness. It has been shown in [4] that ND-based methods are likely to generate oscillatory motion. Moreover, they may cause unreasonable deviation towards free areas, increasing the total distance and time required to execute the mission. Several gap-based approaches have been proposed to deal with these limitations such as [57,58] by performing a tangential-based navigation, or [59,60] by considering the opening angle of the gap. In [4], an enhancement of [57] has been proposed so that the robot safely and smoothly navigates among closely spaced obstacles. Another variant [61], Follow the Gap Method (FGM) considers the field of view constraint. In [62], the length and smoothness of the trajectories generated by FGM were enhanced. Integrating the FGM with the DWA resulted in safer and faster trajectories [63]. Despite the fact that gap-based techniques enable robots to travel through narrow spaces in dense environments, they assume a holonomic and disc-shaped robot, and thus rely on approximations. In order to achieve safe navigation, it is essential to account for the vehicle shape and motion constraints. Addressing this issue is the motivation of the work presented in this paper.

3. Preliminary definitions

The goal and current robot locations are denoted by $\mathfrak{p}_g$ and $\mathfrak{p}_r$, respectively. The robot considered here utilizes a differential-drive (unicycle model), is navigating on a flat ground, and subject to "rolling without slipping" velocity constraints. Its radius (we mean the radius of a virtual circle wrapped around the robot) is denoted by R. The shape of the robot is approximated by a polygon with m edges, denoted as $\mathcal{P}_e$, $e = 1, \ldots, m$.

We assume that the sensor data is available as depth (scan) points to keep the sensor as general as possible; the information acquired by most of the sensors can be processed and reduced to points. The list of scan points is denoted by $S = \{\mathbf{p}_1^S, \ldots, \mathbf{p}_n^S\}$ and ordered counterclockwise with respect to the sensor coordinate system (as they appear in a laser scanner). The polar and Cartesian coordinates of $\mathbf{p}_i^S$ are denoted by (r_i^S, θ_i^S) and (x_i^S, y_i^S), respectively, and the maximum sensor range is denoted by $r_{\max}$.

The order of the scan points in S determines their relative location. For example, point $\mathbf{p}_2^S$ is to the *left* of $\mathbf{p}_1^S$ and point $\mathbf{p}_1^S$ is to the *right* of $\mathbf{p}_2^S$. The sequence of points $\{\mathbf{p}_{i+1}^S, \ldots, \mathbf{p}_n^S\}$ is denoted by $\mathbf{p}_i^{S+}$ and represents those points that are located to the left of $\mathbf{p}_i^S$. Similarly, $\mathbf{p}_i^{S-} = \{\mathbf{p}_{i-1}^S, \ldots, \mathbf{p}_1^S\}$ represents the sequence of points that are located to the right of $\mathbf{p}_i^S$. If the used sensor has a full field of view (FFOV), the sequence of left and right points can be extended beyond $\mathbf{p}_n^S$ and $\mathbf{p}_1^S$, respectively, and therefore, they are modified as follows: $\mathbf{p}_i^{S+} = \{\mathbf{p}_{i+1}^S, \ldots, \mathbf{p}_n^S, \mathbf{p}_1^S, \mathbf{p}_2^S, \ldots, \mathbf{p}_{i-1}^S\}$, $\mathbf{p}_i^{S-} = \{\mathbf{p}_{i-1}^S, \ldots, \mathbf{p}_1^S, \mathbf{p}_n^S, \mathbf{p}_{n-1}^S, \ldots, \mathbf{p}_{i+1}^S\}$. Notice that $\mathbf{p}_i^{S+}$ and $\mathbf{p}_i^{S-}$ have the same elements but the order is reversed. The left (resp. right) neighborhood of a point $\mathbf{p}_i^S$ is denoted as $\mathbf{p}_{i+}^S$ (resp. $\mathbf{p}_{i-}^S$), for instance, the left and right neighborhoods of point $\mathbf{p}_1^S$ are $\mathbf{p}_2^S$ and $\mathbf{p}_n^S$, respectively (assuming a FFOV sensor).

The order of elements in a list (e.g. $\mathbf{p}_i^{S+}$, $\mathbf{p}_i^{S-}$) is of great importance in our approach. It is assumed that the elements are accessed in the same order as they appear in the list.

In order to normalize an angle θ to the range $[-\pi, \pi[$, a projection function is defined as:

$$\text{proj}(\theta) = ((\theta + \pi) \bmod 2\pi) - \pi. \tag{1}$$

Let $\mathcal{F}$ be a frame whose origin is c and whose orientation, with respect to the robot's reference frame, is given by $\theta_{\mathcal{F}}$. The location of any point $\mathbf{p}_i$ in the workspace, relative to frame $\mathcal{F}$ can be expressed by the following equation:

$$^{\mathcal{F}}\mathbf{p}_i = \mathcal{R}^{-1}(\mathbf{p}_i - c) \tag{2}$$

where $\mathcal{R}$ is given by:

$$\mathcal{R} = \begin{bmatrix} \cos(\theta_{\mathcal{F}}) & -\sin(\theta_{\mathcal{F}}) \\ \sin(\theta_{\mathcal{F}}) & \cos(\theta_{\mathcal{F}}) \end{bmatrix}. \tag{3}$$

Assume that $a < b$, a saturation function is defined as follows:

$$\text{sat}_{[a,b]}(x) = \begin{cases} a, & \text{if } x \le a \\ x, & \text{if } a < x < b \\ b, & \text{if } x \ge b. \end{cases} \tag{4}$$

For clarity, the superscripts are removed from figures, e.g. $\mathbf{p}_i^S$ is abbreviated as $\mathbf{p}_i$.

4. Extracting gaps

In this section, we show how to determine the list of gaps visible from the current robot location. The key idea is to analyze the structure of S and find out discontinuities that may occur between two consecutive depth points. The process of detecting and classifying depth discontinuities is discussed in Section 4.1. Section 4.2 describes the first step of the algorithm which implies searching S to extract all visible gaps V surrounding the robot. In the second step, V is reduced to G by eliminating improper or useless gaps as explained in Section 4.3.

4.1. Depth discontinuities

A depth discontinuity corresponds to a region in the workspace that is not visible from the current robot's location. Let $w_{\min}$ be the minimum width of an opening that the robot fits through. Identifying $w_{\min}$ is based on the robot's shape. For example, if a robot has a circular shape $w_{\min}$ is set to its diameter, but for a

rectangular robot, $w_{\min}$ is set to its width. A depth discontinuity occurs between two contiguous scan points $(\mathbf{p}_i^{\mathrm{S}}, \mathbf{p}_j^{\mathrm{S}})$ if one of the following two conditions is fulfilled:
(1) The Euclidean distance between $\mathbf{p}_i^{\mathrm{S}}$ and $\mathbf{p}_j^{\mathrm{S}}$ exceeds $w_{\min}$:

$$\left\|\mathbf{p}_i^{\mathrm{S}} - \mathbf{p}_j^{\mathrm{S}}\right\| > w_{\min}.$$

(2) Either $\mathbf{p}_i^{\mathrm{S}}$ or $\mathbf{p}_j^{\mathrm{S}}$ is a non-obstacle point (corresponds to the maximum sensor range):

$$(r_i^{\mathrm{S}} = r_{\max}) \oplus (r_j^{\mathrm{S}} = r_{\max}).$$

The first discontinuity type has two endpoints (obstacle points) whilst the other has only one. The former is referred to as *a bilateral discontinuity*, while the other is *a unilateral discontinuity*. Define the *basis* of a unilateral discontinuity as its unique endpoint. For a bilateral discontinuity, the basis is the endpoint closer to $\mathfrak{p}_r$.

Every discontinuity is classified as *right* or *left* based on which of its both sides the occluded region is as seen from $\mathfrak{p}_r$. A unilateral discontinuity is a right discontinuity if its basis is to the right of the non-obstacle point. Otherwise, it is a left discontinuity. Analogously, a bilateral discontinuity is classified as right if its basis is to the right of the other endpoint, and classified as a left discontinuity otherwise.

As an example, look at Fig. 1 where we have 4 obstacles, perceived by the robot as 5 lists of points labeled O_1, O_2, O_3, O_4, and O_5. There are two right discontinuities ($\mathrm{d}_{\mathrm{r}1}$, $\mathrm{d}_{\mathrm{r}2}$) and two left discontinuities ($\mathrm{d}_{\mathrm{l}1}$, $\mathrm{d}_{\mathrm{l}2}$) in this figure. $\mathrm{d}_{\mathrm{r}2}$ and $\mathrm{d}_{\mathrm{l}2}$ are of type bilateral, whereas $\mathrm{d}_{\mathrm{r}1}$ and $\mathrm{d}_{\mathrm{l}1}$ are of type unilateral.

4.2. Gaps search

The list of points S is searched for gaps in terms of depth discontinuities. The search process is performed twice: first, by detecting right discontinuities while traveling counterclockwise from point $\mathbf{p}_1^{\mathrm{S}}$ to $\mathbf{p}_n^{\mathrm{S}}$, and then by detecting left discontinuities while traveling clockwise from point $\mathbf{p}_n^{\mathrm{S}}$ to $\mathbf{p}_1^{\mathrm{S}}$. Next, we explain the details of both searches:

Counterclockwise search: Each two contiguous depth points $(\mathbf{p}_i^{\mathrm{S}}, \mathbf{p}_{i+}^{\mathrm{S}})$ are checked for an existence of a *right* discontinuity.[2] The basis of the discontinuity creates the right side of a gap g, denoted by $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$. The procedure of determining the left side $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$ is described in the following.

(1) Let O^+ be the list of *obstacle* points falling to the left of $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$, such that the angular distance traveled does not exceed π:

$$\mathrm{O}^+ = \left\{\mathbf{p}_k^{\mathrm{S}} \in \mathbf{p}_{\mathrm{r}}^{\mathrm{S}+}(\mathrm{g}) \,\middle|\, r_k^{\mathrm{S}} \neq r_{\max},\ \mathrm{proj}(\theta_k^{\mathrm{S}} - \theta_{\mathrm{r}}(\mathrm{g})) > 0\right\} \tag{5}$$

where $\theta_{\mathrm{r}}(\mathrm{g})$ is the angle towards $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$.

(2) Each of the obstacle points contained in O^+ (e.g. $\mathbf{p}_i \in \mathrm{O}^+$) receives a label of *valid* or *invalid*, based on whether the gap created by $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ and $\mathbf{p}_i$ is visible as seen from $\mathfrak{p}_r$ or not.[3] Here, *visible gap* means that the line segment connecting $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ to $\mathbf{p}_i$ does not intersect an obstacle or an occluded region.[4] To do so, we define an angle associated with every point $\mathbf{p}_i \in \mathrm{O}^+$, called the *visibility angle* of $\mathbf{p}_i$ with respect to $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$, as:

$${}^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_i}(\mathrm{g}) = \arccos\left(\frac{r_{\mathrm{r}}^2(\mathrm{g}) + D_i^2 - r_i^2}{2D_i \,.\, r_{\mathrm{r}}(\mathrm{g})}\right) \tag{6}$$

where $r_{\mathrm{r}}(\mathrm{g})$ is the distance to $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$, r_i the distance to $\mathbf{p}_i$, and D_i the Euclidean distance between $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ and $\mathbf{p}_i$.

[2] For a sensor having a limited field of view LFOV, the last two successive points are $\mathbf{p}_{n-1}^{\mathrm{S}}$, $\mathbf{p}_n^{\mathrm{S}}$, but for a FFOV sensor, they are $\mathbf{p}_n^{\mathrm{S}}$, $\mathbf{p}_1^{\mathrm{S}}$.

[3] The subscript in $\mathbf{p}_i$ denotes the index of $\mathbf{p}_i$ relative to O^+, not to S.

[4] $\overline{\mathbf{p}_{\mathrm{r}}(\mathrm{g})\mathbf{p}_i}$ intersects an obstacle or an occluded area if an obstacle falls in the region bounded by the line segments $\overline{\mathfrak{p}_r\mathbf{p}_{\mathrm{r}}(\mathrm{g})}$, $\overline{\mathfrak{p}_r\mathbf{p}_i}$, and $\overline{\mathbf{p}_i\mathbf{p}_{\mathrm{r}}(\mathrm{g})}$.

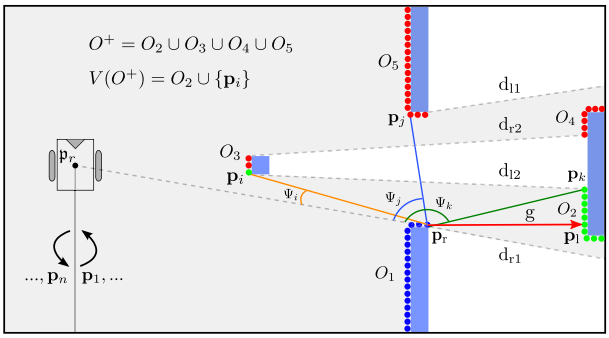

Fig. 1. Detecting a gap g by the counterclockwise search. The sensor data is shown by light gray color, where the solid line indicates the first/final rays. The blue regions are obstacles where the small circles, visualized by blue, green, and red, depict the list of returned scan points S. If a sensor ray hits the boundary of the figure, the maximum sensor range is returned (a non-obstacle point). For clarity, the notation representing the gap, (g), and the subscript $\mathbf{p}_{\mathrm{r}}$ in the visibility angle are removed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

A point $\mathbf{p}_i$ is valid if the following condition is met:

$${}^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_i}(\mathrm{g}) < \wedge^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_{i-}}(\mathrm{g})$$

where $\wedge^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_{i-}}(\mathrm{g})$ is the minimum visibility angle among those corresponding to the points falling in O^+ and coming before $\mathbf{p}_i$:

$$\wedge^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_{i-}}(\mathrm{g}) = \underset{\mathbf{p}_k \in \mathrm{O}^+}{\operatorname{argmin}}\, {}^{\mathbf{p}_{\mathrm{r}}}\Psi_{\mathbf{p}_k}(\mathrm{g}), \quad k < i. \tag{7}$$

(3) Among the valid obstacle points contained in O^+, denoted by $\mathrm{V}(\mathrm{O}^+)$, the one closest to $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ is selected:

$$\mathbf{p}_{\mathrm{l}}(\mathrm{g}) = \underset{\mathbf{p}_k}{\operatorname{argmin}} \left\|\mathbf{p}_k - \mathbf{p}_{\mathrm{r}}(\mathrm{g})\right\|, \quad \mathbf{p}_k \in \mathrm{V}(\mathrm{O}^+). \tag{8}$$

In case of an empty $\mathrm{V}(\mathrm{O}^+)$, $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$ is set to a virtual point at a distance equals to $R+\mathrm{d}_{\mathrm{safe}}$ measured from $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ and an angle equals to $\theta_{\mathrm{r}+}^{\mathrm{S}}(\mathrm{g})$, where $\mathrm{d}_{\mathrm{safe}}$ is a suitable clearance to obstacles and $\theta_{\mathrm{r}+}^{\mathrm{S}}(\mathrm{g})$ the angle towards $\mathbf{p}_{\mathrm{r}+}^{\mathrm{S}}(\mathrm{g})$ (left neighborhood of $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$). Searching for other gaps is resumed starting from $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$. The list of gaps found by the counterclockwise search is denoted by V_{cc}.

In Fig. 1, the basis of $\mathrm{d}_{\mathrm{r}1}$ creates the right side of gap g (at point $\mathbf{p}_{\mathrm{r}}$). $\mathrm{O}^+ = O_2 \cup O_3 \cup O_4 \cup O_5$ is shown by green and red colors to indicate valid and invalid points, respectively. For instance, the point marked as $\mathbf{p}_i$ is valid since its visibility angle, Ψ_i, is less than those corresponding to all points contained in O^+ and coming before $\mathbf{p}_i$ (O_2 in this example). Point $\mathbf{p}_j$ is invalid since $\Psi_j \not< \wedge\Psi_{j-}$ where $\wedge\Psi_{j-} = \Psi_i$ here. It is clear that the line segments connecting $\mathbf{p}_{\mathrm{r}}$ to the invalid points (e.g. $\overline{\mathbf{p}_{\mathrm{r}}\mathbf{p}_j}$) intersect either an occluded region or an obstacle. The left side of g ($\mathbf{p}_{\mathrm{l}}$) is the valid point contained in O^+ and closest to $\mathbf{p}_{\mathrm{r}}$.

Clockwise search: Similar to the counterclockwise search, but performed in the reverse order: we search about *left* discontinuities occurring between each two successive depth points,[5] $(\mathbf{p}_i^{\mathrm{S}}, \mathbf{p}_{i-}^{\mathrm{S}})$. The left side of a gap g, $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$, is created by the basis of the discontinuity. The right side $\mathbf{p}_{\mathrm{r}}(\mathrm{g})$ is identified as follows.

(1) First, we identify the list of *obstacle* points falling to the right of $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$, such that the angular distance traveled $< \pi$:

$$\mathrm{O}^- = \left\{\mathbf{p}_k^{\mathrm{S}} \in \mathbf{p}_{\mathrm{l}}^{\mathrm{S}-}(\mathrm{g}) \,\middle|\, r_k^{\mathrm{S}} \neq r_{\max},\ \mathrm{proj}\,(\theta_k^{\mathrm{S}} - \theta_{\mathrm{l}}(\mathrm{g})) < 0\right\} \tag{9}$$

where $\theta_{\mathrm{l}}(\mathrm{g})$ is the angle towards $\mathbf{p}_{\mathrm{l}}(\mathrm{g})$.

[5] For a sensor having a LFOV, the last two successive points are $\mathbf{p}_2^{\mathrm{S}}$, $\mathbf{p}_1^{\mathrm{S}}$, but for a FFOV sensor, they are $\mathbf{p}_1^{\mathrm{S}}$, $\mathbf{p}_n^{\mathrm{S}}$.

(2) The list of valid obstacle points $V(O^-)$, among those contained in O^-, is identified as follows:

$$V(O^-) = \left\{ \mathbf{p}_i \in O^- \;\middle|\; {}^{\mathbf{p}_l}\Psi_{\mathbf{p}_i}(g) < \wedge^{\mathbf{p}_l}\Psi_{\mathbf{p}_{i-}}(g) \right\} \tag{10}$$

where $\wedge^{\mathbf{p}_l}\Psi_{\mathbf{p}_{i-}}(g)$ is defined by:

$$\wedge^{\mathbf{p}_l}\Psi_{\mathbf{p}_{i-}}(g) = \underset{\mathbf{p}_k \in O^-}{\operatorname{argmin}} {}^{\mathbf{p}_l}\Psi_{\mathbf{p}_k}(g), \quad k < i. \tag{11}$$

(3) The right side of gap g is the obstacle point closest to $\mathbf{p}_l(g)$ and falling in $V(O^-)$:

$$\mathbf{p}_r(g) = \underset{\mathbf{p}_k}{\operatorname{argmin}} \|\mathbf{p}_k - \mathbf{p}_l(g)\|, \quad \mathbf{p}_k \in V(O^-). \tag{12}$$

For an empty $V(O^-)$, $\mathbf{p}_r(g)$ is set to a virtual point at a distance equals to $R+d_{safe}$ measured from $\mathbf{p}_l(g)$ and an angle equals to $\theta^S_{l-}(g)$, where $\theta^S_{l-}(g)$ is the angle towards $\mathbf{p}^S_{r-}(g)$. The search process is then resumed starting from $\mathbf{p}_r(g)$. We denote the list of gaps found by the clockwise search as V_c. Finally, the set of all visible gaps found by both searches is given by $V = V_{cc} \cup V_c$.

4.3. Gaps reduction

The AG method only considers those gaps that are directly facing the robot and discard others. Hence, if a gap leads to another gap, only the first one is taken into account.

Each of the gaps contained in V is classified as *front* or *rear* based on its location relative to the sensor frame. A gap g is a front gap if $|\theta_l(g) - \theta_r(g)| \leq \pi$ and a rear gap otherwise. We say that gap g_j is *reached from* gap g_i if they are of the same type and g_j is *located inside* g_i. If both gaps are of type front, g_j is located inside g_i if the following condition is fulfilled:

$$\theta_r(g_j) \geq \theta_r(g_i) \wedge \theta_l(g_j) \leq \theta_l(g_i).$$

But for rear gaps, the condition to be checked is:

$$\hat{\theta}_r(g_j) \geq \hat{\theta}_r(g_i) \wedge \hat{\theta}_l(g_j) \leq \hat{\theta}_l(g_i)$$

where $\hat{\theta}_x(g)$ denotes the angle directly away from $\theta_x(g)$, and $\theta_x(g)$ represents the angle towards a particular side of gap g:

$$\hat{\theta}_x(g) = \operatorname{proj}(\theta_x(g) - \pi). \tag{13}$$

Now from V every gap that can be reached from another gap is removed. We denote the set of remaining gaps by G:

$$G = V \setminus \left\{ g_j \in V \;\middle|\; g_j \text{ is reached from } g_i \in V,\ g_i \neq g_j \right\}. \tag{14}$$

Fig. 2 shows an example of the complete algorithm: At first, the right discontinuities AB, HI, NO, RS, and WX are detected in the counterclockwise search. Their bases (A, H, N, R, W) create the right sides of the gaps labeled 1–5, respectively. The left sides of these gaps are labeled C, L, Q, V, and Z. In the clockwise search, gaps 6–10 are created by the bases of the left discontinuities UT, QP, LK, FE, and ZY, and the points R, M, G, D, and W, respectively. Gaps 2, 3, 6, and 10 can be reached from gaps 8, 7, 4, and 5. Therefore, they are discarded.

Remark 1 (*Comparison to Other Mechanisms*)**.** The total number of gaps extracted by the AG approach is, like the CG method [56], small when compared with those from [1,54,55,2] (ND methods). For instance, look at Fig. 3 where the AG and CG methods return 5 and 4 gaps labeled 1–5 and *i*–*iv*, respectively, while the ND methods detect 14 gaps marked as A–N (discontinuities in our approach). As compared with the CG method, our approach avoids incorrect or inappropriate gaps like the one labeled *ii* in Fig. 3. It is clear that gaps 2 and 3, that lead to gap *ii*, are more reasonable to be detected in such a case.[6] Moreover, gaps *iii* and *iv* have been replaced by gap 5 due to the fact that our approach, unlike the CG, copes with FFOV sensor types. The reason behind detecting gaps 1 and 4 instead of gap *i* is that the AG method always limits the angular width of gaps to less than π. This is necessary to cope with our obstacle avoidance method explained hereinafter in Section 6.

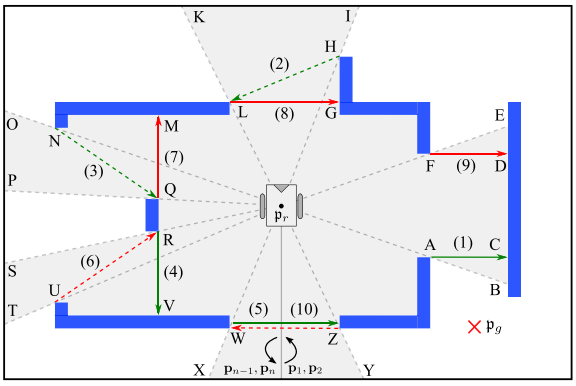

Fig. 2. Extracting gaps. If a ray hits the boundary of the figure, no obstacle point is returned (e.g. at I and K). The gaps extracted by the counterclockwise and clockwise searches are shown by green and red colors, respectively. In the gap reduction process, the gaps indicated by dashed arrows are removed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

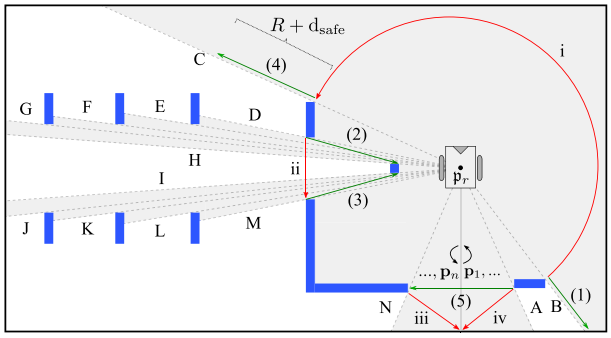

Fig. 3. An example shows the differences between the AG approach and other methods in detecting gaps. The AG detects 5 gaps labeled 1–5, visualized by green arrows. The CG method [56] returns 4 gaps labeled *i*–*iv*, visualized by red arrows. The ND methods (e.g [1] and [2]) extract 12 gaps labeled A–N, equivalent to discontinuities in our approach. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Admissible gap

This section presents the *Admissible gap* concept, which provides a foundation to develop our navigation method. In Section 5.1 we review the robot's kinematic constraints and characterize some properties of the vehicle paths. Section 5.2 introduces a strategy of traversing gaps that complies with the kinematic constraints. In Section 5.3 we use this strategy to determine the admissibility status of a given gap. In order to enhance the readability in this section, the notation representing the gap will be removed, e.g. $\mathbf{p}_l(g)$ is abbreviated as $\mathbf{p}_l$.

[6] Even though gap *ii* can be traversed by the robot, it will be considered as non-navigable since the line segment connecting the robot to its center passes through an obstacle. The AG method avoids this problem by checking the validity of the obstacle points while determining the second side of the gap.

5.1. Kinematic constraints

Every mobile robot is subject to a variety of constraints that may limit its motion. As mentioned in Section 3, we assume a differential-drive robot. It has been shown in the literature that the motion of such a robot is constrained by [64]:

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \tag{15}$$

where (x, y, θ) represents the robot's configuration (location and orientation) relative to the world coordinates.

The kinematic model of this robot can be expressed by:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta\\ \sin\theta\\ 0\end{bmatrix} v + \begin{bmatrix}0\\ 0\\ 1\end{bmatrix} w \tag{16}$$

where v and w are the linear and angular velocities.

In [35], it has been shown that, under the assumption of piecewise constant velocities, the trajectory of a robot whose kinematic model is given by Eq. (16) can be approximated by a sequence of circular arcs whose centers lie on the robot's y-axis [7]. Let $\mathbf{p}_i$ be a point in the workspace, the robot moves along a circular path, denoted as $\mathcal{T}_i$, to reach $\mathbf{p}_i$. The radius of $\mathcal{T}_i$ is referred to as r_i and given by:

$$r_i = \frac{x_i^2 + y_i^2}{2y_i}, \quad r_i \in]-\infty, \infty[\tag{17}$$

where x_i and y_i are the Cartesian coordinates of $\mathbf{p}_i$. The instantaneous center of curvature of $\mathcal{T}_i$ is $(0, r_i)$ and denoted as c_i.

Fig. 4 shows two points, $\mathbf{p}_1$ and $\mathbf{p}_2$, where the robot moves along circles $\mathcal{T}_1$ and $\mathcal{T}_2$ to reach them. The radius and center of $\mathcal{T}_1$ (resp. $\mathcal{T}_2$) are labeled $r1$ and c_1 (resp. $r2$ and c_2).

As we will see in Section 5.2, it is necessary to describe the direction of travel along $\mathcal{T}_i$. For this purpose, a *tangent direction* that uniquely describes $\mathcal{T}_i$ is defined as follows:

$$\chi_i = \begin{cases} \arctan\left(\dfrac{1}{r_i}\right), & \text{if } x_i \geq 0 \\ \operatorname{sgn}(y_i)\cdot\pi - \arctan\left(\dfrac{1}{r_i}\right), & \text{otherwise.}\end{cases} \tag{18}$$

The sign of x_i distinguishes forward from backward motion. This definition can be seen as the direction towards $\mathbf{p}_i$, taking into account the kinematic constraints.

When the robot is at point $\mathbf{p}_i$, its orientation θ_i is tangent to $\mathcal{T}_i$ and given by (see θ_1 and θ_2 in Fig. 4):

$$\theta_i = \begin{cases} \arccos\left(\dfrac{r_i - y_i}{r_i}\right), & \text{if } \operatorname{sgn}(x_i) = \operatorname{sgn}(y_i) \\ -\arccos\left(\dfrac{r_i - y_i}{r_i}\right), & \text{otherwise.}\end{cases} \tag{19}$$

The distance to $\mathbf{p}_i$ along $\mathcal{T}_i$ is the length of the arc that connects $(0, 0)$ to $\mathbf{p}_i$ (see s_1 and s_2 in Fig. 4):

$$s_i = \begin{cases} |x_i|, & \text{if } y_i = 0 \\ |\theta_i \cdot r_i|, & \text{otherwise.}\end{cases} \tag{20}$$

Let $\mathbf{p}_j$ be any point in the workspace, the point on $\mathcal{T}_i$ closest to $\mathbf{p}_j$ is defined by:

$$\mathbf{p}_j(\mathcal{T}_i) = \langle 0, r_i\rangle + \vec{\mathbf{u}}_a \cdot |r_i| \tag{21}$$

where $\vec{\mathbf{u}}_a = \vec{\mathbf{a}}/|\vec{\mathbf{a}}|$ is the unit direction vector of $\vec{\mathbf{a}} = \langle x_j, y_j - r_i\rangle$ and (x_j, y_j) the Cartesian coordinates of $\mathbf{p}_j$. In Fig. 4 the point on $\mathcal{T}_2$ closest to point $\mathbf{p}_3$ is labeled $\mathbf{p}_3(\mathcal{T}_2)$.

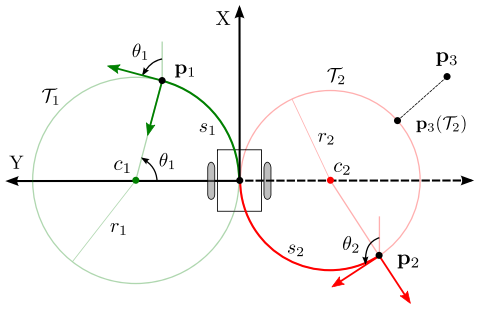

Fig. 4. Geometry of vehicle paths that obey the kinematic constraints. The paths leading to points $\mathbf{p}_1$ and $\mathbf{p}_2$ are described by circles $\mathcal{T}_1$ and $\mathcal{T}_2$, whose radii and centers are labeled $r1$, c_1, $r2$, and c_2, respectively. When the robot reaches points $\mathbf{p}_1$ and $\mathbf{p}_2$, its orientation is θ_1 and θ_2 and the distance traveled is s_1 and s_2. $\mathbf{p}_3(\mathcal{T}_2)$ is the point on $\mathcal{T}_2$ closest to $\mathbf{p}_3$.

5.2. Kinematically constrained gap traversal

There are several ways in which a gap $g \in G$ can be traversed, e.g. across its center. Our strategy here is to drive the robot through g in such away that a compromise between the path length and the motion safety is achieved. A subgoal, denoted as $\mathbf{p}_s$, is assigned to g so that the robot moves along the boundary of one of its sides $\mathbf{p}_c$. When circumnavigating $\mathbf{p}_c$, the robot tends to maintain a suitable distance d_s to it, while simultaneously making progress towards g. Setting d_s is based on the width of the gap; in case of a narrow g, d_s is set to its half width, but in case of a wide g, d_s is limited to $R + d_{safe}$, so that the resultant trajectory is shorter:

$$d_s = \begin{cases} R + d_{safe}, & \text{if } w(g) > 2(R + d_{safe}) \\ \dfrac{1}{2}w(g), & \text{otherwise}\end{cases} \tag{22}$$

where $w(g) = \|\mathbf{p}_l - \mathbf{p}_r\|$.

Identifying $\mathbf{p}_c$ is based on the location of g with respect to the goal and robot configurations. Let $\mathbf{p}_{cg}$ be the gap side closer to the goal and $\mathbf{p}_m$ the midpoint between $\mathbf{p}_r$ and $\mathbf{p}_l$. Since our objective is to make progress towards the goal, $\mathbf{p}_c$ is set to $\mathbf{p}_{cg}$. However, if this makes the robot getting closer than d_s from either $\mathbf{p}_l$ or $\mathbf{p}_r$, the side closer to the robot $\mathbf{p}_{cr}$ along $\mathcal{T}_m$ (a virtual path leading to $\mathbf{p}_m$) is chosen instead:

$$\mathbf{p}_c = \begin{cases} \mathbf{p}_{cg}, & \text{if } \|\mathbf{p}_l(\mathcal{T}_m) - \mathbf{p}_l\| > d_s \wedge \|\mathbf{p}_r(\mathcal{T}_m) - \mathbf{p}_r\| > d_s \\ \mathbf{p}_{cr}, & \text{otherwise}\end{cases} \tag{23}$$

where $\mathbf{p}_l(\mathcal{T}_m)$ and $\mathbf{p}_r(\mathcal{T}_m)$ are the points on $\mathcal{T}_m$ closest to $\mathbf{p}_l$ and $\mathbf{p}_r$, respectively (see Eq. (21)), and $\mathbf{p}_{cr}$ is defined by:

$$\mathbf{p}_{cr} = \begin{cases} \mathbf{p}_l, & \text{if } s_l(\mathcal{T}_m) \leq s_r(\mathcal{T}_m) \\ \mathbf{p}_r, & \text{otherwise}\end{cases} \tag{24}$$

where $s_l(\mathcal{T}_m)$ and $s_r(\mathcal{T}_m)$ are the lengths of the arcs along $\mathcal{T}_m$, connecting $(0, 0)$ to $\mathbf{p}_l(\mathcal{T}_m)$ and $\mathbf{p}_r(\mathcal{T}_m)$ (see Eq. (20)).

Let $\mathcal{S}$ be a circle centered at $\mathbf{p}_c$ with radius d_s, $\mathbf{p}_s$ is located at the tangent point $\mathbf{p}_t$ between $\mathcal{S}$ and the circular trajectory $\mathcal{T}_t$ leading to $\mathbf{p}_t$. The circles $\mathcal{T}_t$ and $\mathcal{S}$ are mutually tangent if:

$$x_c^2 + (y_c - r_t)^2 = (|r_t| \pm d_s)^2 \tag{25}$$

where x_c and y_c are the Cartesian coordinates of $\mathbf{p}_c$, relative to the robot's reference frame and r_t the radius of $\mathcal{T}_t$.

Solving for r_t, it can be shown that two circles are tangent to $\mathcal{S}$, whose radii are defined by:

$$r_t = \frac{x_c^2 + y_c^2 - d_s^2}{2(y_c \pm d_s)}. \tag{26}$$

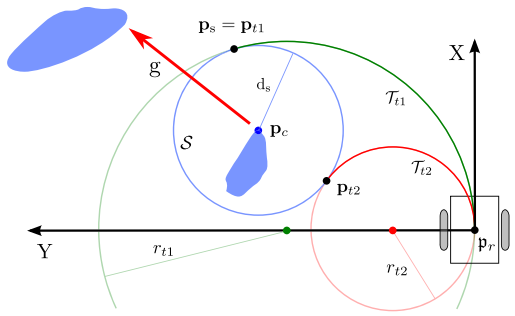

Fig. 5. This figure shows how a subgoal $\mathbf{p}_s$ is assigned to a gap g such that the robot moves along the boundary of its side $\mathbf{p}_c$ while keeping a safe distance d_s to it and respecting the kinematic constraints. First, we locate two points, $\mathbf{p}_{t1}$ and $\mathbf{p}_{t2}$, each of which is tangential to circle S centered at $\mathbf{p}_c$ with radius d_s, and to another circle whose center lies on the robot's y-axis ($\mathcal{T}_{t1}$, $\mathcal{T}_{t2}$). Then, $\mathbf{p}_s$ is set to the tangent point leading to g, $\mathbf{p}_{t1}$.

Given r_t, $\mathbf{p}_t$ can be defined as follows:

$$\mathbf{p}_t = \begin{cases} \langle x_c, 0\rangle, & \text{if } r_t = \infty \\ \langle 0, r_t\rangle + \vec{\mathbf{u}}_v \cdot |r_t|\,, & \text{otherwise} \end{cases} \tag{27}$$

where $\vec{\mathbf{u}}_v = \vec{\mathbf{v}}/|\vec{\mathbf{v}}|$, $\vec{\mathbf{v}} = \langle x_c, y_c - r_t\rangle$. Notice that r_t is ∞ if $y_c = -d_s$. In this case, $\mathcal{S}$ is tangent to the x-axis of the robot, i.e. $\mathcal{T}_t$ is a line, not a circle.

From Eqs. (26) and (27) we get two tangent points, $\mathbf{p}_{t1}$ and $\mathbf{p}_{t2}$. The subgoal $\mathbf{p}_s$ is set to the tangent point that is located in the direction leading to the gap. We use the *tangent direction* defined in Eq. (18) to determine which one it is:

$$\mathbf{p}_s = \begin{cases} \mathbf{p}_{t1}, & \text{if } \text{proj}\,(\chi_{t1} - \chi_c) \cdot \Upsilon < 0 \\ \mathbf{p}_{t2}, & \text{if } \text{proj}\,(\chi_{t2} - \chi_c) \cdot \Upsilon < 0 \end{cases} \tag{28}$$

where χ_t and χ_c are the tangent directions associated with $\mathcal{T}_t$ and $\mathcal{T}_c$ (a virtual trajectory leading to $\mathbf{p}_c$), and Υ is defined by:

$$\Upsilon = \begin{cases} +1, & \text{if } \mathbf{p}_c \text{ is a left side} \\ -1, & \text{if } \mathbf{p}_c \text{ is a right side}\,. \end{cases} \tag{29}$$

Fig. 5 shows how the subgoal $\mathbf{p}_s$ corresponding to gap g is determined. As we can see, two circles $\mathcal{T}_{t1}$ and $\mathcal{T}_{t2}$ are mutually tangent to circle S at $\mathbf{p}_{t1}$ and $\mathbf{p}_{t2}$. The subgoal $\mathbf{p}_s$ is set to $\mathbf{p}_{t1}$, since it is located in the direction leading to g.

Remark 2 (*Eliminating Oscillations*)**.** Locating the subgoal $\mathbf{p}_s$ at $\mathbf{p}_t$ on circle $\mathcal{S}$ guarantees a safe behavior as a distance of d_s is maintained to $\mathbf{p}_c$. However, what if the robot's origin is already located on $\mathcal{S}$, i.e. $\|\mathbf{p}_r - \mathbf{p}_c\| = d_s$. In this case, the value of $\mathbf{p}_t$ from Eq. (27) is (0, 0), causing the robot to turn on spot. Furthermore, what if the robot gets inside $\mathcal{S}$ while circumnavigating $\mathbf{p}_c$, i.e. $\|\mathbf{p}_r - \mathbf{p}_c\| < d_s$. Here, $\mathbf{p}_s$ causes the robot to move away from $\mathbf{p}_c$, until it gets outside $\mathcal{S}$, and the process repeats once the robot gets inside $\mathcal{S}$ again. In the following, we show how to overcome these situations. Let $\mathcal{C}$ be a circle centered at $\mathbf{p}_c$ with radius $r_c = \|\mathbf{p}_r - \mathbf{p}_c\|$. Once $\mathbf{p}_r$ gets on or inside $\mathcal{S}$, $\mathbf{p}_s$ is located on $\mathcal{C}$ rather than on $\mathcal{S}$. The length l of the arc connecting $\mathbf{p}_r$ to $\mathbf{p}_s$ can be of any value, we set $l = r_c \cdot \frac{\pi}{4}$. By this means, the current distance to $\mathbf{p}_c$ is maintained. Of course, we could have two points on $\mathcal{C}$ at a distance of l. As above, the one leading to the gap is selected, see Eq. (28).

5.3. Gap admissibility status

Obstacle avoidance consists of generating a collision-free motion for the next time interval, while making progress towards the goal. The result is a sequence of motion commands, computed at each control cycle, that drives a robot from an initial location towards a target while avoiding collisions. Within this context, only those paths that are the consequence of executing a single motion command are considered.

Assume that the execution of a constant control $u = (v, w)$ results in a path that passes through $\mathbf{p}_s$, described above by $\mathcal{T}_s$. Let $\tau\,[\mathbf{p}_r \to \mathbf{p}_s]$ be the arc the robot follows along $\mathcal{T}_s$ to reach $\mathbf{p}_s$. A gap g is admissible from $\mathbf{p}_r$ ($\text{AG}\,[\mathbf{p}_r \to \text{g}]$) iff $\tau\,[\mathbf{p}_r \to \mathbf{p}_s]$ is collision free,[7] i.e. the robot does not intersect any obstacle in the workspace while traveling along $\tau\,[\mathbf{p}_r \to \mathbf{p}_s]$:

$$\text{AG}\,[\mathbf{p}_r \to \text{g}] \iff \mathcal{T}\,[\mathbf{p}_r \to \mathbf{p}_s] \cap \bigcup_{i=1}^{n} \mathbf{p}_i^S = \phi$$

where $\mathcal{T}\,[\mathbf{p}_r \to \mathbf{p}_s]$ denotes the subset of the workspace occupied by the robot while moving along $\tau\,[\mathbf{p}_r \to \mathbf{p}_s]$.

Next, we introduce a procedure to determine the admissibility status of a given gap g for any vehicle shape. The algorithm extracts the set of obstacle points causing collision with the robot while traveling along $\tau\,[\mathbf{p}_r \to \mathbf{p}_s]$, denoted $\mathcal{O}\,[\mathbf{p}_r \to \mathbf{p}_s]$.

For each obstacle point $\mathbf{p}_i^S \in$ S we check whether the circle centered at $c_s = (0, r_s)$ and passes through $\mathbf{p}_i^S$, denoted by $\mathcal{C}(c_s, \mathbf{p}_i^S)$, intersects any of the robot's edges or not.[8] If not, $\mathbf{p}_i^S$ is discarded, e.g. circle $\mathcal{C}(c_s, \mathbf{p}_3)$ in Fig. 6 does not hit the robot's boundary, and therefore $\mathbf{p}_3$ is discarded (collision-free). However, if an intersection occurs (e.g. circle $\mathcal{C}(c_s, \mathbf{p}_1)$ in Fig. 6), we do the following: Let $\mathbf{p}_e$ be the point of intersection with edge $\mathcal{P}_e$ and (x_s, y_s) the Cartesian coordinates of $\mathbf{p}_s$, relative to the robot's reference frame. The new location of $\mathbf{p}_e$ when the robot reaches $\mathbf{p}_s$, traveling along $\tau\,[\mathbf{p}_r \to \mathbf{p}_s]$, is given by:

$$\mathbf{p}_e^* = \begin{cases} \mathbf{p}_e + \mathbf{p}_s, & \text{if } y_s = 0 \\ \mathcal{R}\,\mathbf{p}_e + t, & \text{otherwise} \end{cases} \tag{30}$$

where $\mathcal{R}$ and t are defined as follows:

$$\mathcal{R} = \begin{bmatrix} \dfrac{x_s^2 - y_s^2}{x_s^2 + y_s^2} & -\dfrac{2x_s y_s}{x_s^2 + y_s^2} \\ \dfrac{2x_s y_s}{x_s^2 + y_s^2} & \dfrac{x_s^2 - y_s^2}{x_s^2 + y_s^2} \end{bmatrix}, t = \begin{bmatrix} x_s \\ y_s \end{bmatrix}. \tag{31}$$

The given points $\mathbf{p}_e$, $\mathbf{p}_e^*$, and $\mathbf{p}_i^S$ all lie on $\mathcal{C}(c_s, \mathbf{p}_i^S)$, it is apparent that $\mathbf{p}_i^S$ is in collision with the robot if it is located on the arc connecting $\mathbf{p}_e$ to $\mathbf{p}_e^*$, considering the direction of travel along $\tau\,[\mathbf{p}_r \to \mathbf{p}_s]$. For checking this condition, we define a frame $\mathcal{F}$ whose origin is c_s and whose x-axis points towards $\mathbf{p}_e$ (visualized by dark blue in Fig. 6), i.e. its orientation is given by:

$$\theta_{\mathcal{F}} = \text{atan2}\,(y_e - r_s, x_e) \tag{32}$$

where (x_e, y_e) are the Cartesian coordinates of $\mathbf{p}_e$.

The obstacle $\mathbf{p}_i^S$ is located on the arc connecting $\mathbf{p}_e$ to $\mathbf{p}_e^*$, and thus added to the output list $\mathcal{O}\,[\mathbf{p}_r \to \mathbf{p}_s]$, if:

$$(\Delta \cdot {}^{\mathcal{F}}\theta_i^S) \bmod 2\pi \le (\Delta \cdot {}^{\mathcal{F}}\theta_e^*) \bmod 2\pi$$

where ${}^{\mathcal{F}}\theta_i^S$ and ${}^{\mathcal{F}}\theta_e^*$ are the angles towards $\mathbf{p}_i^S$ and $\mathbf{p}_e^*$, relative to frame $\mathcal{F}$ (see Eq. (2)), and Δ is given by:

$$\Delta = \begin{cases} +1, & \text{if } \text{sgn}(x_s) = \text{sgn}(y_s) \\ -1, & \text{otherwise.} \end{cases} \tag{33}$$

[7] It may occur that $\tau\,[\mathbf{p}_r \to \mathbf{p}_s]$ intersects the line segment connecting both sides of g, i.e. the subgoal is located inside g. In such a case, the collision check is performed along the path to the intersection point $\mathbf{p}_x$ rather than to $\mathbf{p}_s$, i.e. along $\tau\,[\mathbf{p}_r \to \mathbf{p}_x]$.

[8] If $\mathbf{p}_s$ is located on the robot's x-axis, $\tau\,[\mathbf{p}_r \to \mathbf{p}_s]$ will be a line segment rather than a circular arc. In this case, we check whether the line passes through $\mathbf{p}_i^S$ and parallel to $\overrightarrow{\mathbf{p}_r \mathbf{p}_s}$ intersects any of the robot's edges or not.

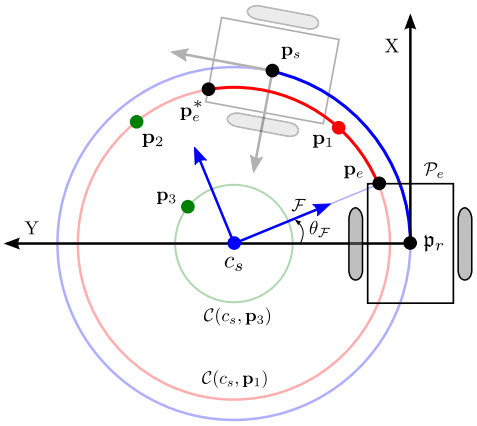

Fig. 6. Collision check along the path towards a subgoal $\mathbf{p}_s$, $\tau\,[\mathfrak{p}_r \rightarrow \mathbf{p}_s]$ (visualized by a dark blue circular arc), given 3 obstacle points $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$. Circle $\mathcal{C}(c_s, \mathbf{p}_1)$ that passes though $\mathbf{p}_1$ and $\mathbf{p}_2$ intersects the robot in $\mathbf{p}_e$. Obstacle $\mathbf{p}_1$ causes collision since it is located on the dark red arc connecting $\mathbf{p}_e$ to $\mathbf{p}_e^*$, but obstacle $\mathbf{p}_2$ does not. Circle $\mathcal{C}(c_s, \mathbf{p}_3)$ does not intersect the boundary of robot. Therefore, obstacle $\mathbf{p}_3$ is collision-free. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Notice that if $\tau\,[\mathfrak{p}_r \rightarrow \mathbf{p}_s]$ is a line segment (i.e. $y_s = 0$), $\mathbf{p}_i^S$ is added to $\mathcal{O}\,[\mathfrak{p}_r \rightarrow \mathbf{p}_s]$ if:

$$\mathrm{sgn}(x_s)x_e \leq \mathrm{sgn}(x_s)x_i^S \leq \mathrm{sgn}(x_s)(x_e^*) \tag{34}$$

where x_e^* is the x-coordinate of $\mathbf{p}_e^*$.

For example, in Fig. 6 obstacle $\mathbf{p}_1$ is located on the arc connecting $\mathbf{p}_e$ to $\mathbf{p}_e^*$ (dark red), which is not the case for $\mathbf{p}_2$. Hence, $\mathbf{p}_1$ causes collision, but $\mathbf{p}_2$ does not.

This procedure is repeated for each edge of polygon $\mathcal{P}_e$. However, if the boundary of the robot can be approximated by a single equation (e.g. a circle), the intersection between $\mathcal{C}(c_s, \mathbf{p}_i^S)$ and the robot's boundary is checked by one step rather than checking each edge of a polygonal robot.

6. AG obstacle avoidance method

In this section, we show how the Admissible Gap concept can be used to navigate a mobile robot. The overall approach works as follows. At each control cycle the sensor data is checked to determine whether the goal is navigable from the current robot's location or not (i.e. $\mathcal{O}\left[\mathfrak{p}_r \rightarrow \mathfrak{p}_g\right] = \phi$). If not, the robot is directed towards a gap rather than towards the goal itself. The gap closest to the goal is selected. It is determined by checking both sides of each gap and selecting the one with the side yielding the minimum Euclidean distance with the goal. The selected gap is then checked for navigability as described in Section 6.1. If it is non-navigable, another gap is selected in the same manner and the process repeats until a navigable gap is found or it is determined that no navigable gap exists. In Section 6.2, we show how the AG method computes the motion commands that guide a robot towards the goal or the selected gap.

6.1. Gap navigability check

Generally speaking, a gap g is *navigable* from $\mathfrak{p}_r$ if the robot can safely pass through it. Hence, an admissible gap is navigable by definition. A non-admissible gap can be navigable if it is possible to find a sequence of *virtual* gaps g_i^*, $i = 1, \ldots, k$, such that g is admissible from g_k^*, g_i^* is admissible from g_{i-1}^*, and g_1^* is admissible from $\mathfrak{p}_r$. However, in the context of obstacle avoidance, it is sufficient to find one virtual gap g^*, such that it is admissible from $\mathfrak{p}_r$ and traversing it leads to gap g. Notice that by constructing ARM in [7], only admissible gaps appear navigable. Next, we introduce an algorithm to determine the location of g^*, if exists. The algorithm takes as an input a gap g and the list of depth points S. The output g^* is initialized to g and determined by iteratively performing two steps:

Step 1: The list of points S is divided into two sublists; the first one contains those points that are located within gap g^*, as seen from $\mathfrak{p}_r$ (i.e. $\{\mathbf{p}_\mathrm{r}(\mathrm{g}^*), \mathbf{p}_{\mathrm{r}+}^S(\mathrm{g}^*), \ldots, \mathbf{p}_{\mathrm{l}-}^S(\mathrm{g}^*), \mathbf{p}_\mathrm{l}(\mathrm{g}^*)\}$). This sublist is called the interior of g^* and denoted by S_in. The other sublist is clearly the relative complement of S_in in S, which is referred to as the exterior of g^* and denoted by S_ex. Non-obstacle points are then removed from S_in and S_ex. The new sublists are referred to as O_in and O_ex. From O_ex, we eliminate every obstacle point that yields an angular distance of more than π with $\mathbf{p}_\mathrm{l}(\mathrm{g}^*)$ and $\mathbf{p}_\mathrm{r}(\mathrm{g}^*)$, while traveling in the direction of the gap. The list of remaining obstacle points is denoted as O'_ex:

$$\mathrm{O}'_\mathrm{ex} = \mathrm{O}_\mathrm{ex}^+ \cup \mathrm{O}_\mathrm{ex}^- \tag{35}$$

with O_ex^+ and O_ex^- given by:

$$\mathrm{O}_\mathrm{ex}^+ = \left\{\mathbf{p}_i^S \in \mathrm{O}_\mathrm{ex} \,\middle|\, \mathrm{proj}\,(\theta_i^S - \theta_\mathrm{r}(\mathrm{g}^*)) > 0\right\} \tag{36}$$

$$\mathrm{O}_\mathrm{ex}^- = \left\{\mathbf{p}_i^S \in \mathrm{O}_\mathrm{ex} \,\middle|\, \mathrm{proj}\,(\theta_i^S - \theta_\mathrm{l}(\mathrm{g}^*)) < 0\right\} \tag{37}$$

where $\theta_\mathrm{r}(\mathrm{g}^*)$ and $\theta_\mathrm{l}(\mathrm{g}^*)$ are the angles towards $\mathbf{p}_\mathrm{r}(\mathrm{g}^*)$ and $\mathbf{p}_\mathrm{l}(\mathrm{g}^*)$.

Each obstacle point $\mathrm{p}_i^S \in \mathrm{O}'_\mathrm{ex}$ is checked for collision with the robot while traveling along $\tau\,[\mathfrak{p}_r \rightarrow \mathbf{p}_\mathrm{s}(\mathrm{g}^*)]$, using the procedure explained in Section 5.3. The outcome is a set of colliding obstacles, denoted as $\mathcal{O}'_\mathrm{ex}\,[\mathfrak{p}_r \rightarrow \mathbf{p}_\mathrm{s}(\mathrm{g}^*)]$. If $\mathcal{O}'_\mathrm{ex}\,[\mathfrak{p}_r \rightarrow \mathbf{p}_\mathrm{s}(\mathrm{g}^*)]$ is not empty, it is clear that g^* is not admissible. However, there is a possibility of creating a new virtual gap (could be admissible) leading to the current one, and therefore, we proceed to step 2 of the algorithm. On the other hand, if $\mathcal{O}'_\mathrm{ex}\,[\mathfrak{p}_r \rightarrow \mathbf{p}_\mathrm{s}(\mathrm{g}^*)]$ is empty, the interior obstacles are checked for collision in the same manner, resulting in $\mathcal{O}_\mathrm{in}\,[\mathfrak{p}_r \rightarrow \mathbf{p}_\mathrm{s}(\mathrm{g}^*)]$, and the algorithm terminates. If $\mathcal{O}_\mathrm{in}\,[\mathfrak{p}_r \rightarrow \mathbf{p}_\mathrm{s}(\mathrm{g}^*)]$ is empty, the output is set to the current value of g^* as it is admissible. Otherwise, it is set to NULL. Notice that the obstacle points removed from O_ex have been excluded from the collision check as they are located behind the robot while traveling along $\tau\,[\mathfrak{p}_r \rightarrow \mathbf{p}_\mathrm{s}(\mathrm{g}^*)]$.

Step 2: In this step a new virtual gap g^{**} is constructed, given g^* and $\mathcal{O}'_\mathrm{ex}\,[\mathfrak{p}_r \rightarrow \mathbf{p}_\mathrm{s}(\mathrm{g}^*)]$. The first side of g^{**}, $\mathbf{p}_\mathrm{f}(\mathrm{g}^{**})$, is created by the obstacle point falling in $\mathcal{O}'_\mathrm{ex}\,[\mathfrak{p}_r \rightarrow \mathbf{p}_\mathrm{s}(\mathrm{g}^*)]$ and closest to the path leading to $\mathbf{p}_\mathrm{s}(\mathrm{g}^*)$, $\mathcal{T}_\mathrm{s}(\mathrm{g}^*)$:

$$\mathbf{p}_\mathrm{f}(\mathrm{g}^{**}) = \underset{\mathbf{p}_i^S}{\mathrm{argmin}} \|\mathbf{p}_i^S - \mathbf{p}_i^S(\mathcal{T}_\mathrm{s}(\mathrm{g}^*))\| \tag{38}$$

where $\mathbf{p}_i^S \in \mathcal{O}'_\mathrm{ex}\,[\mathfrak{p}_r \rightarrow \mathbf{p}_\mathrm{s}(\mathrm{g}^*)]$ and $\mathbf{p}_i^S(\mathcal{T}_\mathrm{s}(\mathrm{g}^*))$ represents the point on $\mathcal{T}_\mathrm{s}(\mathrm{g}^*)$ closest to $\mathbf{p}_i^S$ (see Eq. (21)).

Let $\mathbf{p}_\mathrm{m}(\mathrm{g}^*)$ be the midpoint between $\mathbf{p}_\mathrm{r}(\mathrm{g}^*)$ and $\mathbf{p}_\mathrm{l}(\mathrm{g}^*)$, we define a frame $\mathcal{M}$ whose origin is (0, 0) and whose x-axis points towards $\mathbf{p}_\mathrm{m}(\mathrm{g}^*)$. Assume that $\mathcal{M}$ divides the workspace into two subspaces; one to the left ($\mathcal{W}^+$) and the other to the right ($\mathcal{W}^-$) of the x-axis. Furthermore, let $\tilde{\mathrm{O}}_\mathrm{ex} = \mathrm{O}_\mathrm{ex} \cup \mathbf{p}_\mathrm{r}(\mathrm{g}^*) \cup \mathbf{p}_\mathrm{l}(\mathrm{g}^*)$. The other side of the gap, $\mathbf{p}_\mathrm{o}(\mathrm{g}^{**})$, is searched for in $\tilde{\mathrm{O}}_\mathrm{ex}$ starting from $\mathbf{p}_\mathrm{r}(\mathrm{g}^*)$ and proceeding clockwise if $\mathbf{p}_\mathrm{f}(\mathrm{g}^{**})$ is located in $\mathcal{W}^+$, or starting from $\mathbf{p}_\mathrm{l}(\mathrm{g}^*)$ and proceeding counterclockwise if $\mathbf{p}_\mathrm{f}(\mathrm{g}^{**})$ is located in $\mathcal{W}^-$. The obstacle point closest to $\mathbf{p}_\mathrm{f}(\mathrm{g}^{**})$ is selected, such that the angular distance between both sides of g^{**}, in the direction of g^*, does not exceed π. This is to ensure that navigation through g^{**} leads to g^*. i.e.:

$$\mathbf{p}_\mathrm{o}(\mathrm{g}^{**}) = \underset{\mathbf{p}_i^S \in \tilde{\mathrm{O}}_\mathrm{ex}}{\mathrm{argmin}} \|\mathbf{p}_i^S - \mathbf{p}_\mathrm{f}(\mathrm{g}^{**})\|, \;\; \gamma < \beta < \pi \tag{39}$$

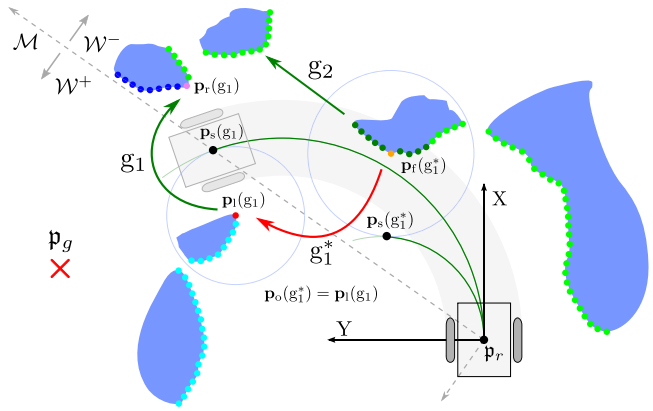

Fig. 7. Gap navigability check. This figure shows two gaps, g_1 and g_2, visible from $\mathfrak{p}_r$, where g_1 is the closest to $\mathfrak{p}_g$. Gap g_1, whose right and left sides are visualized by violet and red dots, is not admissible from $\mathfrak{p}_r$ since the path leading to $\mathbf{p}_s(g_1)$ is in collision with the robot's boundary. However, it is navigable, since a virtual gap g_{1^*} could be constructed, such that it is admissible from $\mathfrak{p}_r$ and traversing it leads to g_1. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

with γ and β defined by:

$$\beta = \begin{cases} {}^{\mathcal{M}}\theta_f(g^{**}) - {}^{\mathcal{M}}\theta_i^S, & \text{if } \mathbf{p}_f(g^{**}) \in \mathcal{W}^+ \\ {}^{\mathcal{M}}\theta_i^S - {}^{\mathcal{M}}\theta_f(g^{**}), & \text{otherwise} \end{cases} \tag{40}$$

$$\gamma = \begin{cases} {}^{\mathcal{M}}\theta_f(g^{**}) - {}^{\mathcal{M}}\theta_r(g^*), & \text{if } \mathbf{p}_f(g^{**}) \in \mathcal{W}^+ \\ {}^{\mathcal{M}}\theta_l(g^*) - {}^{\mathcal{M}}\theta_f(g^{**}), & \text{otherwise} \end{cases} \tag{41}$$

where ${}^{\mathcal{M}}\theta_f(g^{**})$, ${}^{\mathcal{M}}\theta_i^S$, ${}^{\mathcal{M}}\theta_r(g^*)$, and ${}^{\mathcal{M}}\theta_l(g^*)$ are the angles towards $\mathbf{p}_f(g^{**})$, $\mathbf{p}_i^S$, $\mathbf{p}_r(g^*)$, and $\mathbf{p}_l(g^*)$ relative to frame $\mathcal{M}$.

Once g^{**} is created, g^* is set to g^{**} and step 1 is repeated. Notice that the algorithm terminates since each time a new virtual gap is created the number of obstacles in O_{ex} is reduced by at least one. If a valid value of g^* is returned, gap g is considered navigable. Otherwise it is not navigable and NULL is returned.

Fig. 7 shows a robot moving towards a goal $\mathfrak{p}_g$, where it can be seen that g_1 is the gap closest to $\mathfrak{p}_g$, and therefore, it is selected to navigate through. The interior of g_1, O_{in}, includes its right $\mathbf{p}_r(g_1)$ and left $\mathbf{p}_l(g_1)$ sides, marked as violet and red, in addition to those obstacle points visualized by dark blue. All other obstacles belong to the exterior of g_1, O_{ex}, where non of them makes an angular distance $> \pi$ with both sides, and therefore, $O'_{ex} = O_{ex}$. It can be seen that the set of colliding obstacles $\mathcal{O}'_{ex}[\mathfrak{p}_r \to \mathbf{p}_s(g_1)]$ includes those points visualized by dark green, in addition to the orange point that is closest to $\mathcal{T}_s(g_1)$, and thus it is set to $\mathbf{p}_f(g_1^*)$. Since $\mathbf{p}_f(g_1^*)$ is located in $\mathcal{W}^-$, $\mathbf{p}_o(g_1^*)$ is searched for in $\tilde{O}_{ex}$ (all points except those visualized by dark blue) starting from $\mathbf{p}_l(g_1)$ and proceeding counterclockwise. The angular distance between both sides must be $< \pi$. Hence, only $\mathbf{p}_l(g_1)$ and the light blue obstacles are valid. Among those points $\mathbf{p}_l(g_1)$ is the closest to $\mathbf{p}_f(g_1^*)$ and thus it is set to $\mathbf{p}_o(g_1^*)$. It is apparent that the path leading to the virtual gap $\tau\left[\mathfrak{p}_r \to \mathbf{p}_s(g_1^*)\right]$ is collision-free. Therefore, the algorithm terminates in the second iteration and returns g_{1^*}.

Remark 3 (*Improving Safety*)**.** The algorithm presented above checks collision with obstacles, considering only the area occupied by the robot (i.e. without any clearance). This may cause the robot to get close to obstacles, and eventually stuck somewhere. A traditional way to overcome this problem is to enlarge the robot's footprint while extracting $\mathcal{O}[\mathfrak{p}_r \to \mathbf{p}_s(g^*)]$. However, this solution has the drawback of considering a gap as non-navigable even though it is not (trade-off between safety and completeness). In the following, a better way to deal with this problem is proposed. First, we apply the algorithm after enlarging the robot's footprint by a suitable clearance (we use a value of $d_s(g^*) - w_{min}$ in our experiments). Each time a new virtual gap is constructed, the obstacle point closest to $\mathcal{T}[\mathfrak{p}_r \to \mathbf{p}_s(g^*)]$ among those contained in $\mathcal{O}[\mathfrak{p}_r \to \mathbf{p}_s(g^*)]$ is recorded. With this step, we can identify the virtual gap that guarantees the maximum clearance g^*_{best}. Then we perform the algorithm again, but with the same robot's size this time (no clearance), starting from g^*_{best} rather than from the original gap g.

6.2. Setting motion commands

In Section 6.1 we have shown how an admissible gap g^* is virtually located in such away that traversing it makes progress towards the original gap g. We have also shown how g^* is determined in an iterative manner, where each new constructed g^* is located between the robot's origin and the old g^*. In principle, a collision free motion is guaranteed by driving the robot towards g^*, i.e. towards the subgoal $\mathbf{p}_s(g^*)$ corresponding to it. Nevertheless, considering all virtual gaps $G^* = \{g_1^*, \ldots g_k^*\}$ which are created in the iterative process may increase the smoothness of the trajectory. This is due to that fact that the complete passage between g and the last constructed g^* is taken into account, providing a kind of lookahead.

Each virtual gap $g_i^* \in G^*$ is weighted to reflect its relative significance. A key idea in determining the weight is the measurement of the risk imposed by driving the robot towards g_i^*. This can be reflected by the *clearance* to obstacles while traveling along $\tau\left[\mathfrak{p}_r \to \mathbf{p}_s(g_i^*)\right]$:

$$\mathrm{cl}\left(g_i^*\right) = \underset{\mathbf{p}_i^S}{\mathrm{argmin}} \| \mathbf{p}_i^S - \mathbf{p}_i^S\left(\mathcal{T}\left[\mathfrak{p}_r \to \mathbf{p}_s\left(g_i^*\right)\right]\right) \| \tag{42}$$

where $\mathbf{p}_i^S\left(\mathcal{T}\left[\mathfrak{p}_r \to \mathbf{p}_s\left(g_i^*\right)\right]\right)$ represents the point falling on $\mathcal{T}\left[\mathfrak{p}_r \to \mathbf{p}_s\left(g_i^*\right)\right]$ and closest to $\mathbf{p}_i^S$.

With this we can now define the weight (clearance measure) corresponding to g_i^* as follows:

$$w(g_i^*) = \begin{cases} 1, & \text{if } \mathrm{cl}_{max} = \mathrm{cl}_{min} \\ \mathrm{sat}_{[0,1]}\left(1 - \dfrac{\mathrm{cl}_{max} - \mathrm{cl}(g_i^*)}{\mathrm{cl}_{max} - \mathrm{cl}_{min}}\right), & \text{otherwise} \end{cases} \tag{43}$$

where cl_{max} and cl_{min} are given by:

$$\mathrm{cl}_{max} = \underset{g_i^* \in G^*}{\mathrm{argmax}}\left(\mathrm{cl}\left(g_i^*\right)\right) \tag{44}$$

$$\mathrm{cl}_{min} = \underset{g_i^* \in G^*}{\mathrm{argmin}}\left(\mathrm{cl}\left(g_i^*\right)\right). \tag{45}$$

It can be deduced from Eq. (43) that the weight increases as the clearance to obstacles, $\mathrm{cl}(g_i^*)$, increases. Notice that the value of $w(g_i^*)$ is 1 (the maximum) if $\mathrm{cl}\left(g_i^*\right) = \mathrm{cl}_{max}$ and 0 (the minimum) if $\mathrm{cl}\left(g_i^*\right) = \mathrm{cl}_{min}$.

The total weight associated with all $g_i^* \in G^*$ is given by:

$$w_{total} = \sum_{i=1}^{k} \left(w(g_i^*)\right)^2. \tag{46}$$

A new subgoal is then defined as the center of gravity (weighted average) of the subgoals corresponding to all g_i^*:

$$\mathbf{P}_s(G^*) = \left(X_s\left(G^*\right), Y_s\left(G^*\right)\right) \tag{47}$$

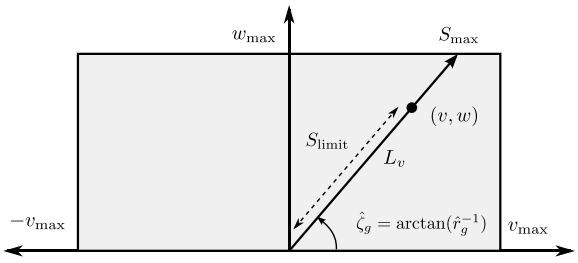

Fig. 8. This figure shows how the motion commands are computed in such a way that a given turning radius $\hat{r}_g$ is preserved, while respecting the maximum possible velocities of the robot. Any motion command (v, w) on line L_v is valid. To maintain safety, (v, w) is selected based on the distance to obstacles. This is achieved by specifying S_{limit}.
Source: Originally from [7].

where X_s (G*) and X_s (G*) are defined by:

$$X_s\left(\mathrm{G}^*\right) = \sum_{i=1}^{k} \frac{\left(w(\mathrm{g}_i^*)\right)^2}{w_{\text{total}}} \cdot x_s(\mathrm{g}_i^*) \tag{48}$$

$$Y_s\left(\mathrm{G}^*\right) = \sum_{i=1}^{k} \frac{\left(w(\mathrm{g}_i^*)\right)^2}{w_{\text{total}}} \cdot y_s(\mathrm{g}_i^*) \tag{49}$$

where $x_s(\mathrm{g}_i^*)$ and $y_s(\mathrm{g}_i^*)$ are the Cartesian coordinates of $\mathbf{p}_s(\mathrm{g}_i^*)$.

The robot is directed towards $\mathbf{P}_s(\mathrm{G}^*)$ if $\tau\,[\mathrm{p}_r \rightarrow \mathbf{P}_s(\mathrm{G}^*)]$ is a collision free path. Otherwise, it is driven towards the subgoal corresponding to the virtual gap (the last constructed g*), i.e. the selected subgoal $\mathbf{p}_s$ is set as follows:

$$\mathbf{p}_s = \begin{cases} \mathbf{P}_s(\mathrm{G}^*), & \text{if } \mathcal{O}\left[\mathrm{p}_r \rightarrow \mathbf{P}_s(\mathrm{G}^*)\right] = \phi \\ \mathbf{p}_s(\mathrm{g}^*), & \text{otherwise.} \end{cases} \tag{50}$$

Up to now we have determined the instantaneous target of the robot, either the goal itself p_g or the selected subgoal $\mathbf{p}_s$. Let $\hat{\mathbf{p}}_g$ be the instantaneous target, we describe next how to compute the motion commands (v, w) that guide a robot towards $\hat{\mathbf{p}}_g$.

Owing to the fact that the AG approach assumes a circular robot path at each time interval, velocities are computed in such a way that the turning radius is preserved. Let $\hat{r}_g$ be the radius of curvature leading to $\hat{\mathbf{p}}_g$ (Eq. (17)), any motion command that satisfies $v = w\hat{r}_g$ is valid. In the control space this can be represented by any point on the line having a slope of $\hat{r}_g^{-1}$ referred to as L_v (see Fig. 8) [7]. Our strategy here is to control the robot's speed based on the distance to nearby obstacles, achieving a safer behavior. For this purpose, we define S_{limit} as follows:

$$S_{\text{limit}} = \left(\sqrt{1 - \text{sat}_{[0,1]}\left(\frac{D_{vs} - r_{\min}}{D_{vs}}\right)}\right) . S_{\max} \tag{51}$$

where $r_{\min}$ is the distance to the closest obstacle, $S_{\max}$ the distance from the origin of the control space to the point of intersection of line L_v with the rectangle of maximum velocities, and D_{vs} a distance parameter that identifies how much the speed is limited. The sat operator is used to cap S_{limit} at 0 if the robot touches an obstacle and at $S_{\max}$ if all obstacles are outside D_{vs}.

With this, the motion commands can be defined as follows:

$$v = \text{sgn}(\hat{x}_g) \cdot S_{\text{limit}} \cdot \cos(\hat{\zeta}_g) \tag{52}$$

$$w = \text{sgn}(\hat{x}_g) \cdot S_{\text{limit}} \cdot \sin(\hat{\zeta}_g) \tag{53}$$

where $\hat{x}_g$ is the x coordinate of $\hat{\mathbf{p}}_g$ and $\hat{\zeta}_g = \arctan(\hat{r}_g^{-1})$. The sign of $\hat{x}_g$ distinguishes forward from backward motion.

7. Experimental results

The AG method was tested using our rescue robot GETbot; a Pioneer 3-AT with a skid-steer drive. The GETbot has a rectangular shape (0.52 × 0.48) and equipped with an on-board computer (2.6 GHz Intel Core i5-3320M processor). The maximum linear and angular velocities are 0.7 m/s and 2.4 rad/s. Two laser scanners were used to sense the environment; the first is a Hokuyo UTM-30LX covering a range of 30 m and having a resolution of 0.25° with a field of view of 270°. The other is a Hokuyo URG-04LX covering a range of 5.6 m and having a resolution of 0.35° with a field of view of 240°. The former is located at the front of the robot while the other is placed in the back. The scan points returned by both laser scanners are merged, generating a *virtual laser scan* with a full field of view.

Several experiments were performed to validate the power of the proposed AG approach. In the following, we outline the results of six experiments carried out in different environments, where the goal and the sensor data were the only information available to the robot in advance. Videos of these experiments are available, see Appendix. The performance of the AG is compared to three state-of-the-art methods; ARM-ND+ [7], the robot navigation stack DWA-A* [65], and TGF [4]. ARM-ND+ is an application of the ARM abstraction layer to the ND+ method [54] as proposed in [7]. DWA-A* is a common open source approach that employs an A* search on a costmap to find a path and uses the DWA [35] to follow the generated path. For constructing the costmap only the front laser scanner is used. The TGF is a recent gap-based method that demonstrated high performance, assuming a holonomic and disc-shaped robot. To maintain safety in narrow spaces, the maximum velocities were limited to (0.5 m/s, 1.0 rad/s). All methods have been implemented using the Robot Operating System (ROS). For the sake of visualization maps of the environments were generated using an open source SLAM algorithm [66]. In this context, we would like to mention that the mapping process depends on how the robot moves, its speed, which areas are scanned, etc. This makes the generated maps slightly different for each method.[9] However, the mapping is stable and the error is very small, as will be shown in the following experiments. Hence, the comparison of the different algorithms is not affected.

Experiment 1: In this experiment the robot had to negotiate a relatively simple arena consisting of randomly distributed boxes, see Fig. 9a. The obstacle course was successfully navigated using all methods. Fig. 9b–e show the trajectories followed by ARM-ND+, DWA-A*, TGF, and AG, respectively. Oscillation in motion can be observed along the path taken by ARM-ND+, e.g. see the path at points 1–3 in Fig. 9b. This limitation is inherited from the ND+ method [1], see [4] for more information. Using the DWA-A*, the robot moved close to obstacles (e.g. A and B in Fig. 9c), a result of following a path generated by an A* algorithm. No significant differences can be seen, when looking at the paths followed by the TGF and AG methods. They both smoothly and safely drove the robot towards the target. The linear and angular velocities were recorded over the course of the experiment. They are plotted against the time in Fig. 9f–i.

Experiment 2: The major difficulty in this experiment was the presence of a large U-shaped structure that the robot had to avoid, see Fig. 10a. Moreover, a relatively narrow curved passage consisting of different obstacle sizes and shapes (labeled P) had to be passed to reach the goal. It is obvious that the trajectory generated by ARM-ND+ was oscillatory, e.g. see points 1–4 in Fig. 10b. Furthermore, the motion was slow as the robot got close to obstacles, e.g. see the trajectory near the points labeled A and D. By running the DWA-A* method, the robot moved relatively fast.

[9] Loosely speaking, the smoother the trajectory the better the generated map.

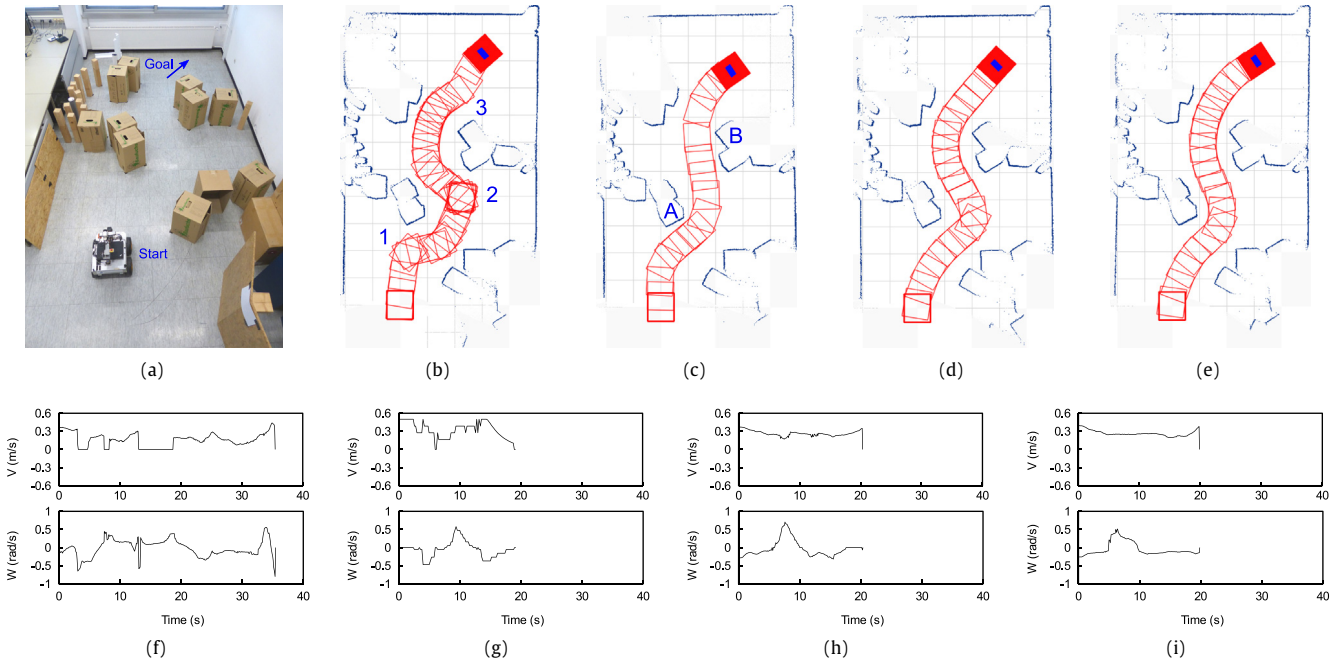

Fig. 9. Experiment 1. (a) Environment setup. (b–e) Trajectories followed by (b) ARM-ND+, (c) DWA-A*, (d) TGF, and (e) AG. (f–i) Velocity profile for (f) ARM-ND+, (g) DWA-A*, (h) TGF, and (i) AG.

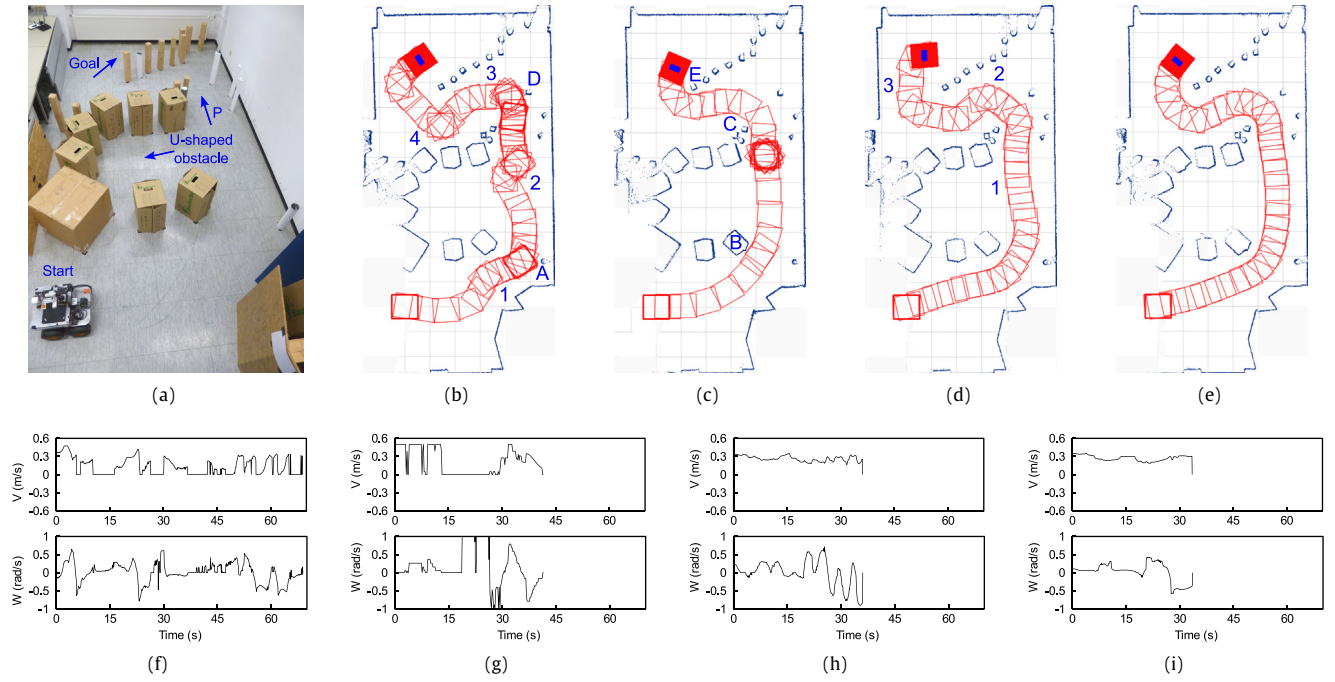

Fig. 10. Experiment 2. (a) Environment setup. (b–e) Trajectories followed by (b) ARM-ND+, (c) DWA-A*, (d) TGF, and (e) AG. (f–i) Velocity profile for (f) ARM-ND+, (g) DWA-A*, (h) TGF, and (i) AG.

However, it got close to obstacles B, C, and E, see Fig. 10c. Moreover, while entering passage P, the robot suddenly stopped and started to rotate in place. After monitoring the output of the algorithm, we found out that this rotation was performed by a recover behavior that was triggered as no valid control could be found at that moment. This behavior attempted to clear space and hopefully free the robot. Fortunately, by executing it, a valid control could be found again, and thus, the robot proceeded towards the goal. The TGF method performed pretty well as can be seen in Fig. 10d. However, the trajectory could be improved at some locations, e.g. at points 1–3. This was possible by applying the AG method that, unlike the TGF, considers the exact shape and kinematics of the robot, see Fig. 10e. The recorded linear and angular velocities are plotted versus time in Fig. 10f–i.

Experiment 3: This experiment had three difficulties. First, a U-shaped obstacle structure had to be avoided. Second, the arena was composed of consecutive narrow curved passages. Finally, the obstacles consisting of small wooden and plastic poles, creating

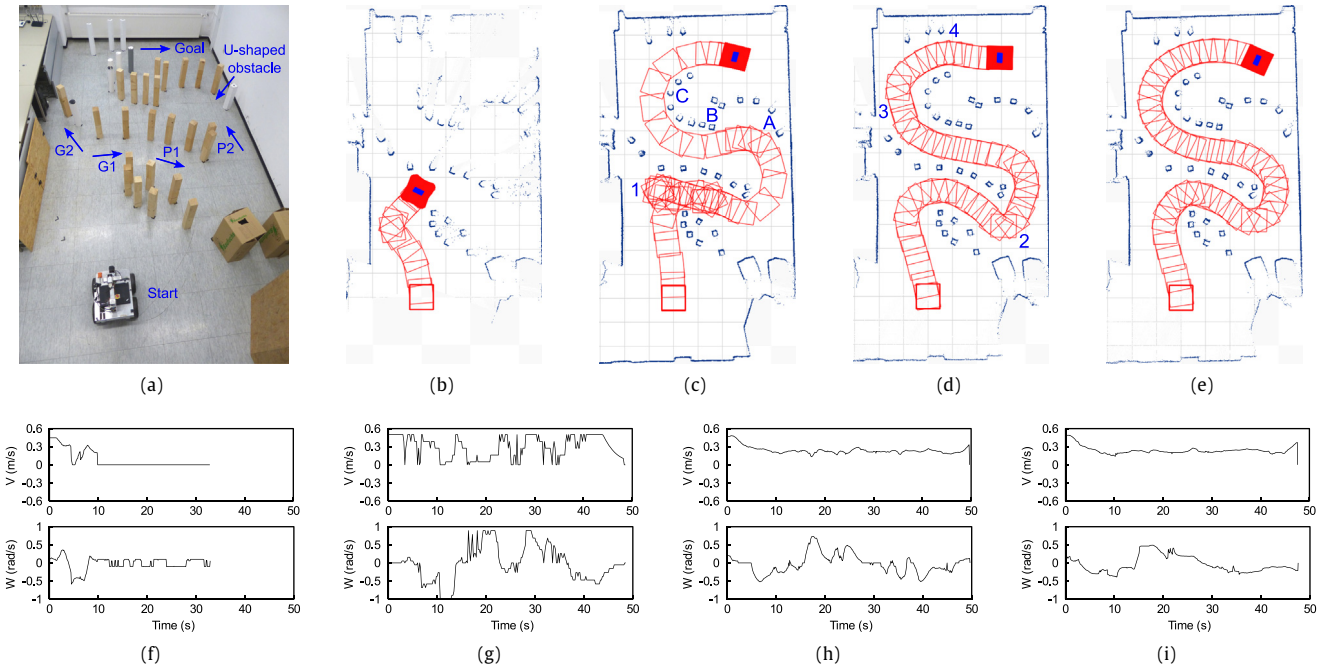

Fig. 11. Experiment 3. (a) Environment setup. (b–e) Trajectories followed by (b) ARM-ND+, (c) DWA-A*, (d) TGF, and (e) AG. (f–i) Velocity profile for (f) ARM-ND+, (g) DWA-A*, (h) TGF, and (i) AG.

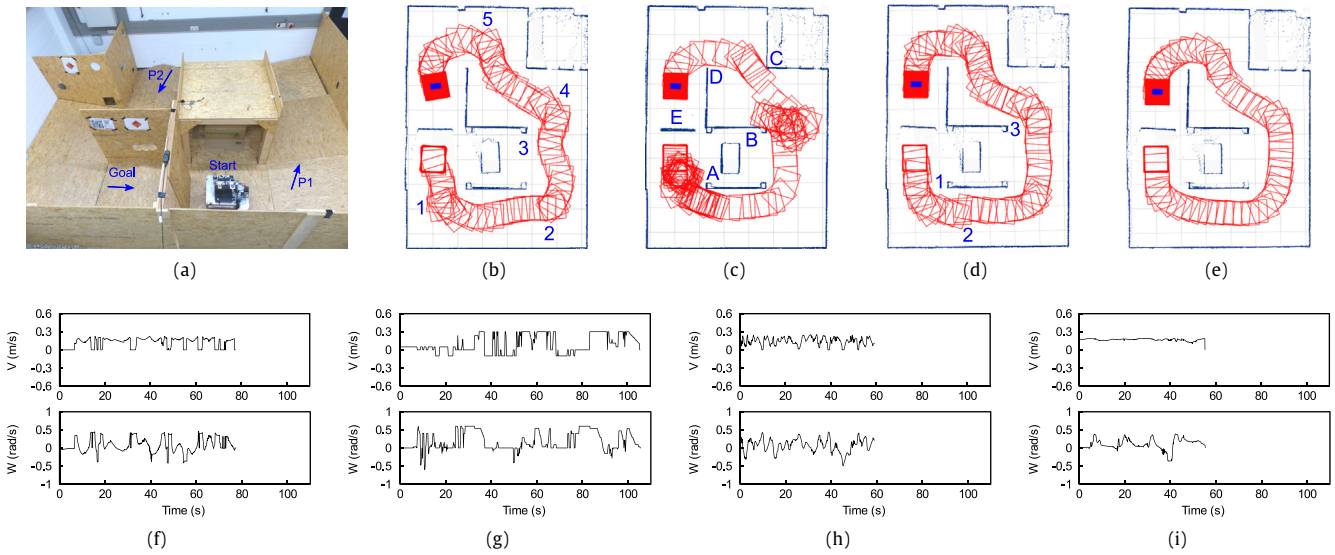

Fig. 12. Experiment 4. (a) Environment setup. (b–e) Trajectories followed by (b) ARM-ND+, (c) DWA-A*, (d) TGF, and (e) AG. (f–i) Velocity profile for (f) ARM-ND+, (g) DWA-A*, (h) TGF, and (i) AG.

openings through which the robot could not pass, see Fig. 11a. Using the ARM-ND+ method, the robot was unable to navigate the course after reaching a location at which it kept turning left and right, see Fig. 11b. We figured out that this behavior occurred due to the disappearance of gap G1 and the detection of the non-navigable gap G2. At that moment, the robot started to turn towards G2 until it had a different view to both gaps, recognizing that G2 is blocked and G1 is free. So, the robot turned again towards G1 and the process repeated. The reason behind the disappearance of gap G1 and the false detection of gap G2 is the usage of the abstraction layer (by constructing ARM only admissible gaps appear navigable). By applying the DWA-A* algorithm, the robot moved nicely until it faced passage P2. At that moment, a sudden change in the costmap occurred, blocking the path planned across P2. Hence, a new path was planned towards the initial position and the robot moved back through passage P1. Once the robot reached the location labeled 1 in Fig. 11c, the blocked area in the costmap appeared free again, and thus, the robot rotated and proceeded towards the goal. The robot got close to obstacles at several locations, e.g. obstacles A–C. The TGF and AG methods smoothly and safely navigated the course and behaved fairly similar except at some locations, where the AG method was slightly better in terms of smoothness. See, for instance, the trajectory near points 2–4 in Fig. 11d and compare it to that shown in Fig. 11e. We provide the velocity profiles for all methods in Fig. 11f–i.

Experiment 4: For this experiment, an environment similar to the RoboCup Rescue Arena was created as shown in Fig. 12a. A smooth navigation reduces the problem of wheel slippage over

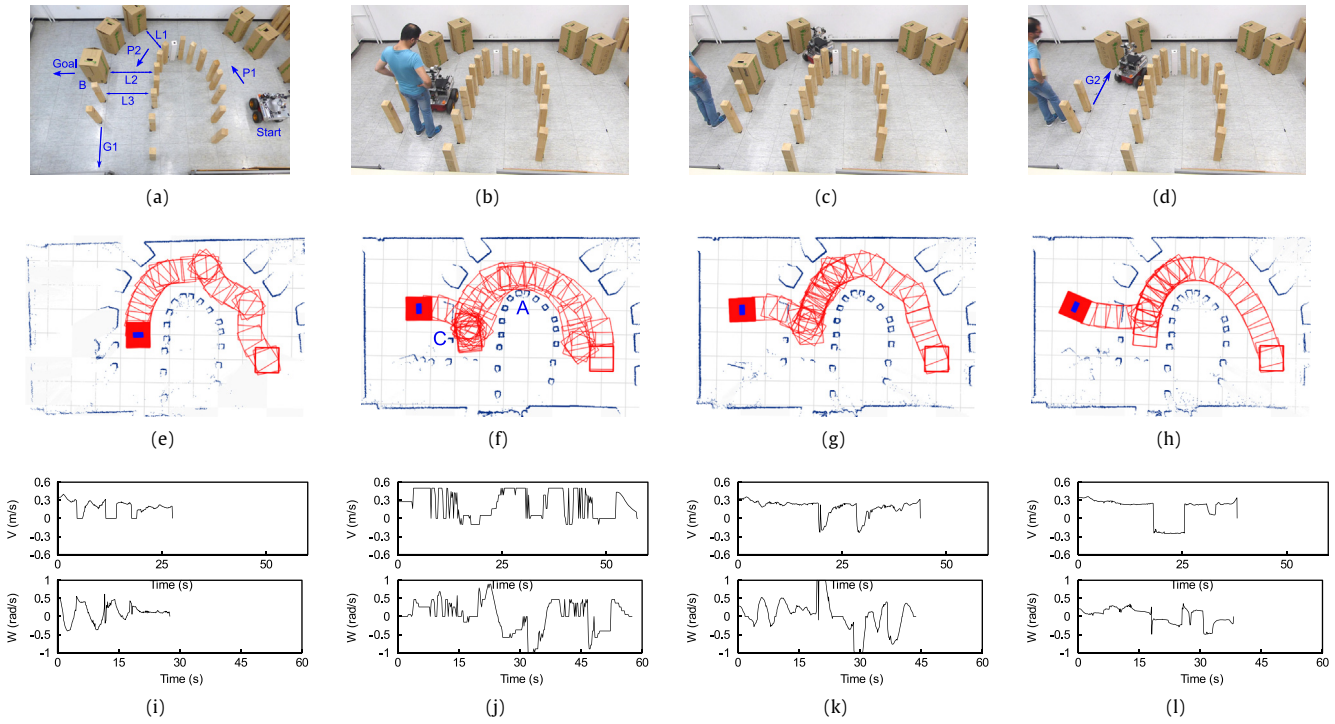

Fig. 13. Experiment 5. (a) Environment setup. (b) a human suddenly closed the passage leading to the goal. (c) The human freed the passage. (d) A box was removed, creating a gap, G2, closer to the goal than gap G1. (e–h) Trajectories followed by (e) ARM-ND+, (f) DWA-A*, (g) TGF, and (h) AG. (i–l) Velocity profile for (i) ARM-ND+, (j) DWA-A*, (k) TGF, and (l) AG.

ramps, and thus of great importance in this scenario. It was risky to move the robot too fast over ramps, especially using DWA-A*. Therefore, the maximum velocities were limited to (0.3 m/s, 0.6 rad/s). The goal was successfully reached using all methods, see Fig. 12b–e. However, using ARM-ND+, the robot suffered from frequent changes in the direction of motion resulting in an oscillatory trajectory, see the path near points 1–5 in Fig. 12b. By running DWA-A*, the robot crashed into the wall marked as B and navigated very close to the walls labeled A, C, and D in Fig. 12c. Moreover, while traveling through the starting area of passages P1 and P2, the robot moved forward and backward for a while before it could overcome the situation and proceed towards the goal. We do believe that this behavior was due to the slippage of the wheels over ramps, causing the robot to move towards the wall and away from the planned local path. Once it happened, a new path had to be planned moving the robot away from the wall, and the process repeated until the robot could escape. It is worth to mention that, at the start of the mission, wall E could not be seen by the front laser scanner, and thus, the robot moved back before discovering that the area behind it was obstructed. Using the TGF method, oscillations in motion are reduced compared to ARM-ND+, but they still exist, e.g. see the trajectory near the points labeled 1–3 in Fig. 12d. It can be observed in Fig. 12e that the AG method could drive the robot much smoother than all previously discussed methods. We confirm our visualization by plotting the velocity profiles in Fig. 12f–i.

Experiment 5: The objective of this experiment was to test the behavior of the AG method in the presence of a dynamic obstacle (a human moving around). In the first part of the experiment, the robot smoothly navigated through the obstacle course shown in Fig. 13a. Once it crossed the line marked as L3, a human stepped in and closed passage P2 (see Fig. 13b). At that moment the robot escaped by moving backwards towards P1, as it was the only passage through which the robot could fit. As soon as the robot crossed line L1, the human stepped back, freeing passage P2 (see Fig. 13c). At the same time, the robot detected the situation and moved forward, proceeding towards the G1 gap. Once the robot crossed line L2, the box labeled B was removed. It was detected that the gap created by removing the box (G2 in Fig. 13d) was navigable and closer to the goal than G1, and thus the robot proceeded through it and reached the goal. The resultant trajectory can be seen in Fig. 13h. By applying ARM-ND+ (Fig. 13e), the robot stopped moving and the mission was aborted, once the human closed passage P2. This is because no navigable gap could be found from that location (ND+ only computes forward motion). Using DWA-A* (Fig. 13f), the reaction, as expected, was slower than that of the AG; the robot touched the feet of the human and took more time to detect gap G2. When the P2 passage was closed, the robot rotated and moved towards P1, even after passage P2 had been freed. Once the complete passage P1 was explored, a new path is set back towards P2. Similar to the previous experiments, the robot got close to obstacles at different locations (e.g. points A and C). By running the TGF method (Fig. 13g), the robot rotated in place and moved forwards rather than backwards, whenever the human closed and freed passage P2. Apart from the reaction to the dynamic obstacle, the increased smoothness of the AG method, compared to all other techniques, is clearly demonstrated in Fig. 13e–h and i–l.

Experiment 6: In this experiment a very difficult and challenging environment was created. Fig. 14a and b show two images of the environment taken from two different viewpoints. The robot had to navigate through extremely narrow and curved passages, where, at some locations, the room available to maneuver was less than the diameter of the robot. For instance, the width of the gap marked as G3 in Fig. 14b was 0.63 m, whereas the diameter of the GETbot is approximately 0.71 m. A second difficulty was the mixture of different size obstacles. Finally, a significant change in clearance was encountered at the gap labeled G1. It was only the AG method that successfully drove the robot towards the goal (Fig. 14f). Although the scenario was very challenging, the motion was extremely smooth. Using ARM-ND+, the robot crashed into the

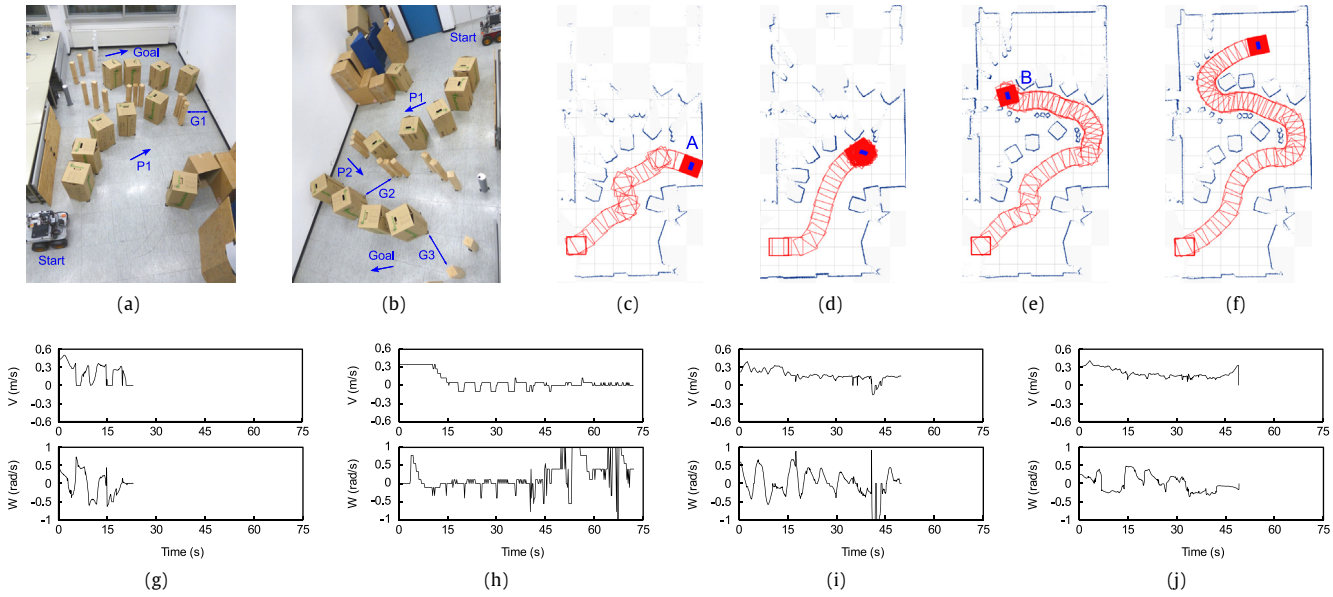

Fig. 14. Experiment 6. (a, b) Environment setup. (c–f) Trajectories followed by (c) ARM-ND+, (d) DWA-A*, (e) TGF, and (f) AG. (g–j) Velocity profile for (g) ARM-ND+, (h) DWA-A*, (i) TGF, and (j) AG.

wall marked as A in Fig. 14c. This is due to the fact that no navigable gap could be found by ARM-ND+ once the robot faced gap G1, and thus the algorithm terminated. Due to the speed acquired before detecting this situation, the robot kept moving until it hit the wall. With the DWA-A* approach, the motion was smooth until the robot reached a point at which it kept moving forward/backward and rotating without being able to pass through gap G1 (Fig. 14d). We figured out that a path was correctly planned towards passage P2, but the local planner (DWA) could not successfully follow it. Using TGF, the robot managed to pass through most of the arena (Fig. 14e). At the end of passage P2, however, gap G3 was considered non-navigable owing to the fact that TGF ignores the robot shape and kinematic constraints; it easily rejects any gap having a width less than 2R. Therefore, the robot rotated in place, trying to escape towards the free area behind it. While rotating, a collision with obstacle B occurred.

8. Evaluation and discussion

In order to evaluate the effectiveness of the AG approach and to compare its performance to that of the techniques discussed in Section 7, the following metrics are employed [4,67]:

(1) Total execution time (T_{tot}): The total amount of time a robot needs to perform the mission.

(2) Path length (P_{len}): The total distance traveled by a robot from an initial location to a target:

$$P_{len} = \int_{x_i}^{x_g} \left(1 + \left(f'(x)\right)^2\right)^{\frac{1}{2}} dx \tag{54}$$

where $(x_i, f(x_i))$ and $(x_g, f(x_g))$ denote the Cartesian coordinates of the initial and target locations.

(3) Average Change of Curvature (C_{avg}): This metric is helpful in detecting oscillations along the trajectory:

$$C_{avg} = \frac{1}{T_{tot}} \int_0^{T_{tot}} \left|k'(t)\right| dt, \quad k(t) = \left|\frac{w(t)}{v(t)}\right| \tag{55}$$

where $(v(t), w(t))$ are the linear and angular robot velocities.

(4) Zero crossings along the angular speed curve (Z_w): This metric is useful in detecting the degree of variation in the steering angle, i.e. turn changes. Therefore, it can be an indication of the stability and integrity in the velocity control.

(5) Accumulated linear and angular jerks (J_{acc}, ζ_{acc}): Jerk, which is the time derivative of acceleration, is associated with sudden changes in the forces exerted by the vehicle actuators. Hence, smoothness in the robot's speed as well as steering can be quantified as a function of jerk:

$$J_{acc} = \frac{1}{T_{tot}} \int_0^{T_{tot}} [\dot{v}(t)]^2 \, dt \tag{56}$$

$$\zeta_{acc} = \frac{1}{T_{tot}} \int_0^{T_{tot}} [\dot{w}(t)]^2 \, dt. \tag{57}$$

(6) Lateral stress (S_{lat}): This metric is related to the vehicle stability as it measures the centrifugal force acting on the robot, integrating it along the trajectory:

$$S_{lat} = \int_0^{T_{tot}} \frac{v(t)^2}{r(t)} \, dt \tag{58}$$

where $r(t)$ is the reciprocal of the curvature.

(7) Tangential stress (S_{tng}): With this metric, it is possible to detect sudden accelerations and decelerations that may lead to wheel slippage on turns:

$$S_{tng} = \int_0^{T_{tot}} |\dot{v}(t)| \, dt. \tag{59}$$

(8) Risk of obstacles (R_{obs}): This metric measures the proximity to obstacles along the entire mission. Let $r_{min}(t)$ be the distance to the closest obstacle point at time t, R_{obs} is given by:

$$R_{obs} = \int_0^{T_{tot}} \frac{1}{r_{min}(t)} \, dt. \tag{60}$$

Notice that for each of the metrics describe above it holds true that, the lower the value, the better the performance. Notice also that a small value is added to the denominator in Eqs. (55), (58), and (60) to avoid division by zero. In our calculations, we used a value of 0.001.

Based on the aforementioned metrics, a performance evaluation of the presented AG approach as well as a comparison to the methods discussed in Section 7 was performed. The experiments

Table 1
Performance evaluation of the proposed AG approach for the experiments presented in Section 7.

Exp.	Method	T_{tot}	P_{len}	C_{avg}	Z_w	J_{acc}	ζ_{acc}	S_{lat}	S_{tng}	R_{obs}
1	ARM-ND+	35.4	5.89	118.62	10	2.28	17.52	0.95	2.90	252.28
	DWA-A*	**19.0**	**4.91**	0.53	**2**	4.87	6.00	**0.56**	2.23	169.34
	TGF	20.3	5.26	0.67	**2**	0.22	0.98	0.91	0.85	134.44
	AG	19.8	5.24	**0.28**	**2**	**0.02**	**0.89**	0.95	**0.55**	**128.29**
2	ARM-ND+	68.8	9.81	181.15	18	6.82	16.44	1.92	9.34	543.02
	DWA-A*	41.3	**8.03**	161.22	**3**	11.05	55.77	**1.16**	4.91	375.81
	TGF	36.0	9.14	1.29	9	0.10	2.57	2.16	1.65	195.97
	AG	**33.5**	8.71	**0.38**	**3**	**0.04**	**1.77**	1.71	**0.76**	**194.19**
3	ARM-ND+	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
	DWA-A*	48.4	12.95	51.52	5	14.42	32.78	2.40	8.96	510.24
	TGF	49.6	11.45	1.97	10	0.09	**2.09**	2.25	1.77	**273.15**
	AG	**47.5**	**11.10**	**0.39**	**3**	**0.04**	2.20	**1.84**	**1.32**	289.68
4	ARM-ND+	77.3	10.40	119.60	32	2.13	12.96	1.42	6.00	340.61
	DWA-A*	105.4	13.67	103.18	26	7.52	20.69	1.55	11.05	17 415.50
	TGF	67.3	10.01	31.65	26	1.02	2.40	1.12	4.16	260.22
	AG	**55.3**	**9.29**	**0.67**	**8**	**0.03**	**1.53**	**1.31**	**0.86**	**253.06**
5	ARM-ND+	NC	NC	NC	NC	NC	NC	NC	NC	NC
	DWA-A*	57.4	13.92	61.17	6	24.04	39.87	2.60	11.92	940.37
	TGF	43.7	8.85	12.39	12	3.55	23.34	2.53	4.49	399.19
	AG	**37.6**	**8.24**	**0.64**	**5**	**3.48**	**4.38**	**1.16**	**2.38**	**209.72**
6	ARM-ND+	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
	DWA-A*	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
	TGF	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
	AG	**49.1**	**9.74**	**0.76**	**7**	**0.28**	**3.50**	**1.84**	**2.23**	**681.60**

shown in Figs. 9–14 yield the results given in Table 1. Maybe the most important result of these experiments is that the ARM-ND+ algorithm presented the worst performance among all tested methods. The only exception was in experiment 4, where the DWA-A* yielded the worst results. We attribute the poor performance of the ARM-ND+ to two reasons. First, by mapping the workspace into ARM, only those gaps that are directly navigable (admissible) from the current robot's location can be seen. Therefore, a navigable gap may appear blocked as occurred in experiments 3 and 6. Second, the obstacle avoidance method itself (ND+) has the tendency to generate oscillatory and unstable motion as has been proven in [4].

It is important to note that it is unfair to directly compare the performance of the DWA-A* to the other tested methods. This is due to the fact that the DWA-A* is a combination of both global and local planners while the other methods are pure reactive. Despite this fact, however, the performances of both the AG and TGF methods were better, particularly in terms of the C_{avg}, J_{acc}, ζ_{acc}, S_{tng}, and R_{obs} measures. This can be explained as follows: the global A* planner computes a path close to obstacles, causing the robot to move near corners along the traversed route, leading to a higher R_{obs} value. Indeed, navigation near corners hides the local environment to the DWA, and in turn, new obstacles may suddenly appear, requiring a sharp change in speed (reflected by the value of J_{acc}) and direction (reflected by the values of C_{avg}, ζ_{acc}, and S_{tng}). The TGF and AG methods, on the other hand, both tend to drive the robot in such away that a suitable clearance is maintained to obstacles (see Section 5.2). Moreover, oscillations are reduced by computing the avoidance maneuver based on the configuration of obstacles located between the robot and the closest gap, providing a kind of look-ahead. In principle, the DWA-A* may generate shorter paths and can drive the robot at higher speeds. This is due to the fact that the path is computed by an A* algorithm, which is short compared to others, and followed by the DWA that considers the dynamic properties of the robot. However, this benefit was canceled in experiments 2 and 3 due to triggering the recover behavior and re-planning the path. In experiment 4, the situation was even worse since moving near walls led to wheel skidding, and in turn, the robot collided with wall B and consumed more time to pass through the starting area of passages P1 and P2.

Comparing the results of the AG approach with those of the TGF method, we observe a clear performance improvement of AG over TGF, particularly in terms of the C_{avg}, Z_w, and J_{acc} metrics. More important, the AG method successfully passed the course of experiment 6 while TGF did not. This is a direct consequence of computing the AG avoidance maneuver such that the vehicle shape and motion constraints are respected. TGF, on the other hand, determines a direction solution that the robot should follow, regardless of the vehicle constraints. Hence, it relies on approximations when applied on a real vehicle. For example, by taking a closer look at the trajectories of the TGF method in Figs. 9–14, it can be seen that the robot often had to stop and turn on spot to face the direction it should follow, rather than following controlled curved trajectories. By this means, the speed had to be reduced and the curvature had to be changed in a short time, resulting in higher C_{avg}, Z_w, and J_{acc} values. Notice that our mobile robot is roughly square. If it had a more complex shape, the TGF would have worse performance and it could fail to drive the robot in more scenarios, especially those requiring high maneuverability. It is also significant to note that aligning the vehicle with the direction solution was not straightforward. We spent quite a lot of time implementing the TGF to achieve this relatively good performance.

Apart from the results discussed above, the AG approach has several advantages over the other tested methods. First of all, it has only two parameters that can easily be determined (d_{safe} and D_{vs}). d_{safe} just defines how far we like the robot to stay away from obstacles, of course whenever possible (trade-off between safety and path length). Hence, it is considered an advantage rather than a drawback. In principle, d_s in Eq. (22) can always be set to $\frac{1}{2}w(g)$ and totally discard d_{safe}, generating longer trajectories in open spaces. D_{vs} determines how much the speed is limited near obstacles. It can be easily tuned by setting it to a relatively large value at the beginning, being conservative, and gradually decreasing it until an acceptable behavior is achieved. It is, however, possible to completely get rid of this parameter by considering the dynamic properties of the robot (see Section 9). Notice that, in all experiments, d_{safe} and D_{vs} were set to $2R$ and 0.9 m, respectively. The TGF and ARM-ND+ both have an additional parameter, D_s, that is a security zone around the robot, once occupied, the trajectory is adjusted based on the relative proximity of obstacles. Tuning D_s is

not straightforward and highly affects the behavior of the robot. The situation is worse in the DWA-A*, since a lot of parameters have to be carefully tuned, consuming much time and effort.

Another advantage of the AG approach, which we have learned from our tests, is its robustness against changes of the environment. It successfully managed all test scenarios, including those presented in Section 7 or others. Moreover, repeating the same experiment, resulted in almost the same performance. The other methods, particularly ARM-ND+, were sensitive to the structure of the environment. Perhaps, the most tedious and time-consuming part of our experiments was to find out scenarios in which all these techniques succeed. The ARM-ND+ method could only work in simple scenarios. The DWA-A* approach followed a different trajectory each time an experiment was repeated. The TGF method was relatively more robust than both ARM-ND+ and DWA-A*; with a proper tuning of D_s, it handled most of the tests. However, repeating the same scenario presented higher differences in the values of the performance measures, compared to AG. Notice that we have been forced to reduce the difficulty of the environments in experiments 1, 2 and 5, so that these methods could succeed.

An additional, yet important aspect of the AG approach is the simplicity of the problem formulation and implementation. For instance, unlike the ARM-ND+, there is no need to construct an abstraction layer, which is non-trivial and may add a computational overhead. Moreover, unlike the TGF, the presented experiments can be replicated by directly implementing the approach without adapting the output to cope with the shape and motion constraints. Finally, unlike the DWA, AG can drive a robot in very difficult and narrow spaces without having to integrate it with a global planner. A higher level planner is only necessary to avoid cyclic loops or global trap situations.

The experiments presented in Section 7 were conducted using a relatively small robot with a rectangular shape. However, the proposed approach is applicable to bigger robots with arbitrary shape, such as those used in industrial and research facilities. This can be deduced from the fact that the AG avoidance maneuver is computed in such away that the exact vehicle shape and kinematic constraints are respected; refer to Section 5.2 which presents a methodology of traversing gaps in a kinematically admissible way. Refer also to Sections 5.3 and 6.1 that explain how this methodology is employed to determine the navigability status of a given gap for any vehicle shape. It is, however, important to note that the dynamic properties of the robot play a significant role when the weight and/or speed of the robot increases considerably. This issue will be investigated in our future work as pointed out in Section 9.

It is also worth to mention that the AG method can be applied to different sensor types, such as a laser scanner, camera, or kinect. This is owing to the fact that the sensory information is assumed to be available as depth points, see Section 3; most of the sensors can be processed and reduced to points. Nevertheless, the inaccuracy of the sensor may affect the behavior of the robot. For example, extracting the exact gap size by using ultrasonic sensors is no longer possible. By this means, it is difficult to achieve good results in highly cluttered environments. However, this is not a limitation of the method, but rather a limitation of the sensor itself.

9. Conclusions and future works

This paper presents a new concept, the admissible gap, for reactive obstacle avoidance. This concept addresses the question of whether it is possible to find a kinematically admissible motion command that safely guides a mobile robot through a given gap. With this concept, it has been possible to develop an obstacle avoidance approach for robots moving in unknown dense environments. Unlike the gap-based techniques, the proposed AG approach is developed in such away that the exact shape and kinematics are taken into account. Hence, it provides a safer, easier to implement, and more accurate solution than that obtained by adapting the output of a method originally designed for holonomic disc-shaped robots. The key idea is the construction of an admissible gap in such away that, when the robot navigates through it, progress towards the goal is achieved. For this purpose, we have proposed a strategy of traversing gaps in a kinematically admissible way, where a compromise between path length and motion safety is achieved. It is important to note that the AG approach is directly applied to the workspace, and thus there is no need to create any other space. This paper also introduces a new methodology of finding out gaps, that is applicable for FFOV and LFOV sensor types. With this methodology, the total number of gaps is reduced by eliminating useless ones, increasing the stability of navigation and alleviating the possibility of oscillation. Experimental results validated the increased efficiency, safety, smoothness, and robustness of the AG, compared to existing state-of-the-art methods.

Future work includes improving the admissible gap concept such that the dynamic properties of the robot are considered, selecting a dynamically feasible trajectory rather than limiting the speed using D_{vs}. By this means, the robot can safely move at higher speeds. Moreover, we get rid of tuning D_{vs}.

Appendix. Index to multimedia extensions

The multimedia extensions to this article are available at: http://getwww.uni-paderborn.de/research/videos/ag.

Ext.	Media	Description
1	Videos	The accompanying videos show scenes of all experiments presented in Section 7.

References

[1] J. Minguez, L. Montano, Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios, IEEE Trans. Robot. Autom. 20 (1) (2004) 45–59.
[2] J.W. Durham, F. Bullo, Smooth nearness-diagram navigation, in: IROS, Nice, France, 2008, pp. 690–695.
[3] M. Mujahed, B. Mertsching, A new gap-based collision avoidance method for mobile robots, in: SSRR, Lausanne, 2016, pp. 220–226.
[4] M. Mujahed, D. Fischer, B. Mertsching, Tangential gap flow (TGF) navigation: A new reactive obstacle avoidance approach for highly cluttered environments, Robot. Auton. Syst. 84 (2016) 15–30.
[5] A. Bemporad, A.D. Luca, G. Oriolo, A. De, Local incremental planning for a car-like robot navigating among obstacles, in: ICRA, Minneapolis, Minnesota, 1996, pp. 1205–1211.
[6] J. Minguez, L. Montano, Robot navigation in very complex, dense, and cluttered indoor/outdoor evironments, in: 15th IFAC World Congress, Barcelona, Spain, 2002, pp. 1–6.
[7] J. Minguez, L. Montano, Extending collision avoidance methods to consider the vehicle shape, kinematics, and dynamics of a mobile robot, IEEE Trans. Robot. 25 (2) (2009) 367–381.
[8] M. Mujahed, B. Mertsching, The admissible gap (AG) method for reactive collision avoidance, in: IEEE International Conference on Robotics and Automation, ICRA, Singapore, 2017, pp. 1916–1921.
[9] J. Minguez, F. Lamiraux, J.-P. Laumond, Springer Handbook of Robotics, Springer International Publishing, 2016, pp. 1177–1202 (Chapter). Motion Planning and Obstacle Avoidance.
[10] M. Mohanan, A. Salgoankar, A survey of robotic motion planning in dynamic environments, Robotics and Autonomous Systems (2017). http://dx.doi.org/10.1016/j.robot.2017.10.011.
[11] J. Borenstein, Y. Koren, The vector field histogram - fast obstacle avoidance for mobile robots, IEEE Trans. Robot. Automat. 7 (3) (1991) 278–288.
[12] F. Fahimi, C. Nataraj, H. Ashrafiuon, Real-time obstacle avoidance for multiple mobile robots, Robotica 27 (2) (2009) 189–198.
[13] B. Kovács, G. Szayer, F. Tajti, M. Burdelis, P. Korondi, A novel potential field method for path planning of mobile robots by adapting animal motion attributes, Robot. Auton. Syst. 82 (2016) 24–34.
[14] R.T. Rodrigues, M. Basiri, A.P. Aguiar, P. Miraldo, Feature based potential field for low-level active visual navigation, in: ROBOT 2017: Third Iberian Robotics Conference. Advances in Intelligent Systems and Computing, Vol 693, Springer, Cham, 2018, pp. 791–800.

[15] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robot. Res. 5 (1) (1986) 90–98.
[16] Y. Koren, J. Borenstein, Potential field methods and their inherent limitations for mobile robot navigation, in: IEEE International Conference on Robotics and Automation, 1991, pp. 1398–1404.
[17] J. Barraquand, J.C. Latombe, Robot motion planning: A distributed representation approach, Int. J. Robot. Res. 10 (6) (1991) 628–649.
[18] A.A. Masoud, A harmonic potential approach for simultaneous planning and control of a generic UAV platform, J. Intell. Robot. Syst. 65 (1–4) (2012) 153–173.
[19] F. Bounini, D. Gingras, H. Pollart, D. Gruyer, Modified artificial potential field method for online path planning applications, in: IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 2017, pp. 180–185.
[20] J. Ren, K.A. McIsaac, R.V. Patel, Modified Newton's method applied to potential field-based navigation for mobile robots, IEEE Trans. Robot. 22 (2) (2006) 384–391.
[21] D. Panagou, Motion planning and collision avoidance using navigation vector fields, in; ICRA, Hong Kong, China, 2014, pp. 2513–2518.
[22] R. Hegde, D. Panagou, Multi-agent motion planning and coordination in polygonal environments using vector fields and model predictive control, in: ECC, Aalborg, Denmark, 2016, pp. 1856–1861.
[23] N. Malone, H.-T. Chiang, K. Lesser, M. Oishi, L. Tapia, Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field, IEEE Trans. Robot. 33 (5) (2017) 1124–1138.
[24] H. Chiang, N. Malone, K. Lesser, M. Oishi, L. Tapia, Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments, in: IEEE International Conference on Robotics and Automation, ICRA, Seattle, USA, 2015, pp. 2347–2354.
[25] V.J. Lumelsky, A.A. Stepanov, Dynamic path planning for a mobile automaton with limited information on the environment, IEEE Trans. Automat. Control 31 (5) (1986) 1058–1069.
[26] A.S. Matveev, M.C. Hoy, A.V. Savkin, A method for reactive navigation of nonholonomic under-actuated robots in maze-like environments, Automatica 49 (5) (2013) 1268–1274.
[27] D. Maravall, J. de Lope, J.P.F. Brea, Visual bug algorithm for simultaneous robot homing and obstacle avoidance using visual topological maps in an unmanned ground vehicle, in: Bioinspired Computation in Artificial Systems - IWINAC, Part II, LNCS 9108, Switzerland, 2015, pp. 301–310.
[28] I. Kamon, E. Rimon, E. Rivlin, Tangentbug: A range-sensor-based navigation algorithm, Int. J. Robot. Res. 17 (9) (1998) 934–953.
[29] A.V. Savkin, M. Hoy, Reactive and the shortest path navigation of a wheeled mobile robot in cluttered environments, Robotica 31 (2) (2013) 323–330.
[30] W. Feiten, R. Bauer, G. Lawitzky, Robust obstacle avoidance in unknown and cramped environments, in: ICRA, CA, USA, 1994, pp. 2412–2417.
[31] R. Simmons, The curvature-velocity method for local obstacle avoidance, in: ICRA, Minnosota, USA, 1996, pp. 3375–3382.
[32] C. Shi, Y. Wang, J. Yang, A local obstacle avoidance method for mobile robots in partially known environment, Robot. Auton. Syst. 58 (5) (2010) 425–434.
[33] M. Seder, I. Petrovic, Dynamic window based approach to mobile robot motion control in the presence of moving obstacles, in: ICRA, Roma, Italy, 2007, pp. 1986–1991.
[34] D.A. de Lima, A.C. Victorino, A hybrid controller for vision-based navigation of autonomous vehicles in urban environments, IEEE Trans. Intell. Transp. Syst. 17 (8) (2016) 2310–2323.
[35] D. Fox, W. Burgard, S. Thrun, The dynamic window approach to collision avoidance, IEEE Robot. Autom. Mag. 4 (1) (1997) 23–33.
[36] M. Hoy, A.S. Matveev, A.V. Savkin, Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey, Robotica 33 (3) (2015) 463–497.
[37] P. Inigo-Blasco, F. Diaz-del-Rio, S. Vicente Diaz, D. Cagigas Muniz, The shared control dynamic window approach for non-holonomic semi-autonomous robots, in: ISR/Robotik, 2014, pp. 1–6.
[38] P. Ogren, N.E. Leonard, A convergent dynamic window approach to obstacle avoidance, IEEE Trans. Robot. 21 (2) (2005) 188–195.
[39] A. Maroti, D. Szaloki, D. Kiss, G. Tevesz, Investigation of Dynamic Window Based Navigation Algorithms on a Real Robot, in: Proceedings of IEEE 11th International Symposium on Applied Machine Intelligence and Informatics, SAMI, 2013, pp. 95–100.
[40] J. Ballesteros, C. Urdiales, A.B.M. Velasco, G. Ramos-Jimenez, A biomimetical dynamic window approach to navigation for collaborative control, IEEE Trans. Hum.-Mach. Syst. 47 (6) (2017) 1123–1133.
[41] A. Wu, J.P. How, Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles, Auton. Robot. 32 (3) (2012) 227–242.
[42] M. Kim, J. Oh, Study on optimal velocity selection using velocity obstacle (OVVO) in dynamic and crowded environment, Auton. Robots 40 (8) (2016) 1459–1470.
[43] B.H. Lee, J.D. Jeon, J.H. Oh, Velocity obstacle based local collision avoidance for a holonomic elliptic robot, Auton. Robots 41 (6) (2017) 1347–1363.
[44] P. Fiorini, Z. Shiller, Motion planning in dynamic environments using velocity obstacles, Int. J. Robot. Res. 17 (7) (1998) 760–772.
[45] T. Fraichard, H. Asama, Inevitable collision states - a step towards safer robots? Adv. Robot. 18 (10) (2004) 1001–1024.
[46] A. Lawitzky, A. Nicklas, D. Wollherr, M. Buss, Determining states of inevitable collision using reachability analysis, in: IROS, Chicago, USA, 2014, pp. 4142–4147.
[47] D. Bareiss, J. Berg, Generalized reciprocal collision avoidance, Int. J. Robot. Res. 34 (12) (2015) 1501–1514.
[48] T.L. Baldi, S. Scheggi, M. Aggravi, D. Prattichizzo, Haptic guidance in dynamic environments using optimal reciprocal collision avoidance, IEEE Robot. Automat. Lett. 3 (1) (2018) 265–272.
[49] E.G. Szadeczky-Kardoss, B. Kiss, Velocity obstacles for Dubins-like mobile robots, in: 25th Mediterranean Conference on Control and Automation, MED, Valletta, Malta, 2017, pp. 346–351.
[50] S. Roelofsen, D. Gillet, A. Martinoli, Collision avoidance with limited field of view sensing: A velocity obstacle approach, in: IEEE International Conference on Robotics and Automation, ICRA, Singapore, 2017, pp. 1922–1927.
[51] B. Gopalakrishnan, A.K. Singh, M. Kaushik, M. Krishna, D. Manocha, Prvo: Probabilistic reciprocal velocity obstacle for multi robot navigation under uncertainty, in: IEEE International Conference on Intelligent Robots and Systems, IROS, Vancouver, Canada, 2017.
[52] J. Jin, Y. Kim, S. Wee, N. Gans, Decentralized cooperative mean approach to collision avoidance for nonholonomic mobile robots, in: ICRA, Seattle, USA, 2015, pp. 35–41.
[53] Z. Shiller, S. Sharma, High speed on-line motion planning in cluttered environments, in: IROS, Vilamoura, Portugal, 2012, pp. 596–601.
[54] J. Minguez, J. Osuna, L. Montano, A "Divide and Conquer" strategy based on situations to achieve reactive collision avoidance in troublesome scenarios, in: ICRA, 2004, pp. 3855–3862.
[55] J. Minguez, The obstacle-restriction method for robot obstacle avoidance in difficult environments, in: IROS, 2005, pp. 2284–2290.
[56] M. Mujahed, D. Fischer, B. Mertsching, H. Jaddu, Closest gap based (CG) reactive obstacle avoidance navigation for highly cluttered environments, in: IROS, Taipei, Taiwan, 2010, pp. 1805–1812.
[57] M. Mujahed, H. Jaddu, D. Fischer, B. Mertsching, Tangential closest gap based (TCG) reactive obstacle avoidance navigation for cluttered environments, in: SSRR, Linköping, Sweden, 2013, pp. 1–6.
[58] M. Mujahed, D. Fischer, B. Mertsching, Robust collision avoidance for autonomous mobile robots in unknown environments, in: 20th International World RoboCup Symposium, Springer Lecture Notes on Artificial Intelligence, Vol. 9776, Leipzig, Germany, 2016, pp. 1–12.
[59] M. Mujahed, D. Fischer, B. Mertsching, Safe gap based (SG) reactive navigation for mobile robots, in: European Conference on Mobile Robots, ECMR, Barcelona, Spain, 2013, pp. 325–330.
[60] M. Mujahed, D. Fischer, B. Mertsching, Smooth reactive collision avoidance in difficult environments, in: IEEE Conference on Robotics and Biomimetics, ROBIO, Zhuhai, China, pp. 1471–1476.
[61] V. Sezer, M. Gokasan, A novel obstacle avoidance algorithm: Follow the gap method, Robot. Auton. Syst. 60 (9) (2012) 1123–1134.
[62] M. Demir, V. Sezer, Improved follow the gap method for obstacle avoidance, in: IEEE International Conference on Advanced Intelligent Mechatronics, AIM, Munich, Germany, 2017, pp. 1435–1440.
[63] A. Oezdemir, V. Sezer, A hybrid obstacle avoidance method: follow the gap with dynamic window approach, in: IEEE International Conference on Robotic Computing, IRC, Taichung, Taiwan, 2017, pp. 257–262.
[64] J. paul Laumond, S. Sekhavat, F. Lamiraux, Guidelines in nonholonomic motion planning for mobile robots, Robot Motion Plann. Control 229 (1998) 1–53.
[65] E. Marder-Eppstein, E. Berger, T. Foote, B.P. Gerkey, K. Konolige, The office marathon: Robust navigation in an indoor office environment, in: ICRA, Alaska, USA, 2010, pp. 300–307.
[66] B. Gerkey, contributors, Karto mapping library, (April 2010). http://wiki.ros.org/karto/ (accessed: 09.02.2017).
[67] D. Calisi, D. Nardi, Performance evaluation of pure-motion tasks for mobile robots with respect to world models, Auton. Robots 27 (4) (2009) 465–481.

Muhannad Mujahed was born on 25 September 1980 in Hebron, Palestine. He received the Computer Engineering degree from Palestine Polytechnic University, Hebron, Palestine, in 2002, and the master degree in Electronics and Computer Engineering from Al-Quds University, Jerusalem, Palestine, in 2010. In 2011, he received a Ph.D. scholarship from the Deutscher Akademischer Austausch Dienst (DAAD) and joined the Cognitive Systems Engineering Group, (GET Lab), University of Paderborn, Paderborn, Germany.

His main areas of interest include mobile robot navigation, collision avoidance, control theory, and sensor-based motion planning.

Dirk Fischer received his diploma degree in an interdisciplinary course of study in Electrical Engineering and Computer Science from the University of Paderborn, Germany in 2000. Since 2003 he is with the cognitive systems engineering group (GET Lab) at the University of Paderborn, Paderborn, Germany. His main research interests include rescue robotics, robot navigation, and motion planning.

Bärbel Mertsching received the Ph.D. degree in electrical engineering from Paderborn University, Paderborn, Germany, with a thesis on knowledge-based image analysis.

She was a Professor of Computer Science at the University of Hamburg, Hamburg, Germany, from 1994 to 2003. In 2003 she joined the University of Paderborn, Paderborn, Germany, where she is currently a Professor of Electrical Engineering and the director of the Cognitive Systems Engineering Group, (GET Lab). Her research interests include cognitive systems engineering, in particular, computer vision and robotics, as well as microelectronics for image processing and didactics of engineering education.