Robotics and Autonomous Systems 84 (2016) 15–30

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

Tangential Gap Flow (TGF) navigation: A new reactive obstacle avoidance approach for highly cluttered environments

Muhannad Mujahed *, Dirk Fischer, Bärbel Mertsching

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

HIGHLIGHTS

- A novel reactive collision avoidance approach, referred to as TGF, is presented.
- The TGF safely drives a mobile robot in very dense and cluttered environments.
- The trajectory is faster, shorter, and smoother compared to the well-known ND method.
- Experimental results demonstrate the power of the TGF approach.
- The performance is evaluated and compared with three different ND variants.

ARTICLE INFO

Article history:
Received 3 September 2015
Accepted 7 July 2016
Available online 18 July 2016

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

ABSTRACT

This paper presents a novel reactive collision avoidance method for mobile robots moving in dense and cluttered environments. The proposed method, entitled Tangential Gap flow (TGF), simplifies the navigation problem using a divide and conquer strategy inspired by the well-known Nearness-Diagram Navigation (ND) techniques. At each control cycle, the TGF extracts free openings surrounding the robot and identifies the suitable heading which makes the best progress towards the goal. This heading is then adjusted to avoid the risk of collision with nearby obstacles based on two concepts namely, *tangential* and *gap flow* navigation. The *tangential navigation* steers the robot parallel to the boundary of the closest obstacle while still emphasizing the progress towards the goal. The *gap flow navigation* safely and smoothly drives the robot towards the free area in between obstacles that lead to the target. The resultant trajectory is faster, shorter and less-oscillatory when compared to the ND methods. Furthermore, identifying the avoidance maneuver is extended to consider all nearby obstacle points and generate an avoidance rule applicable for all obstacle configurations. Consequently, a smoother yet much more stable behavior is achieved. The stability of the motion controller, that guides the robot towards the desired goal, is proved in the Lyapunov sense. Experimental results including a performance evaluation in very dense and complex environments demonstrate the power of the proposed approach. Additionally, a discussion and comparison with existing Nearness-Diagram Navigation variants is presented.

1. Introduction

The increasing need for employing robots in high-risk areas hit by natural disasters has attracted the attention of researchers worldwide to develop fully autonomous mobile robots. The main objective of these robots is to carry out the assigned tasks in places where human presence is dangerous, difficult or the tasks to be performed are impossible to be carried out by people [1], such as search and rescue, military and exploration.

Usually, a real world disaster environment is partially or completely unknown and changes over time. Moreover, unpredictable obstacles may block the robot's trajectory while performing tasks. Therefore, traditional motion planning techniques which depend on a predefined map cease to function properly in such environments and the robot is doomed to collide with obstacles. To overcome this limitation the mobile robot should have the capability to explore the unknown area, automatically generate maps, and localize itself within this map. This is achieved by on-board sensing devices detecting instantaneous changes in the environment. In addition, it has to recognize victims and correctly label the map

* 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).

http://dx.doi.org/10.1016/j.robot.2016.07.001
0921-8890/

with the victims' positions. In all cases, it is necessary to have a reactive obstacle avoidance algorithm to safely reach given goal locations and to deal with dynamic changes such as moving obstacles.

Many existing reactive navigation methods have problems in dealing with dense and cluttered environments, which is usually the case in most robotic applications. This is due to the fact that these methods suffer from several well-known problems [2] such as local trap situations, difficulties of computing motion directions towards obstacles or far away from the goal location, and the failure of driving the robot between closely spaced obstacles. Navigation in these environments whilst avoiding such problems has been addressed by the Nearness Diagram (ND) Navigation method [2]. Over the years, several ND variants have been developed such as the Obstacle-Restriction method [3], the Smooth Nearness-Diagram (SND) [4], and the Closest Gap (CG) Navigation [5].

Nevertheless, all these techniques share the idea of analyzing sensory data to find potential gaps surrounding the robot, and identifying the direction of motion accordingly. The resultant trajectory is then deflected to avoid collisions with nearby obstacles using an idea inspired from the Artificial Potential Field concept [6]. Obstacle avoidance based on potential fields is likely to produce oscillatory robot trajectories. Such oscillations slow down the overall behavior of the mobile robot and may lead to unstable motion in narrow passages [7]. Moreover, the decoupling between the location of the goal/gap and the direction of the avoidance maneuver increases the problem. The Tangential Gap Flow (TGF) navigation method proposed in this paper is especially designed to deal with this limitation. In a nutshell, the TGF method works as follows: the direct path towards the goal is checked for navigability. If it is a collision-free path, the robot is directly driven towards the goal. Otherwise, the goal is located within the navigable gap closest to it. As soon as the distance to an obstacle gets less than a predefined security distance, the TGF method modifies the trajectory based on two concepts: the *tangential* and *gap flow* navigation. Using the *tangential navigation*, the robot navigates parallel to the tangent of the closest obstacle while simultaneously progresses towards the goal. The *gap flow navigation* safely and smoothly drives the robot towards the free area in between obstacles which lead at the end to the target. The trajectory generated by the TGF method is faster, shorter, and less-oscillatory compared to the ND variants. The stability of the motion controller, that drives the robot towards the goal, is proved in the Lyapunov sense.

The *tangential navigation* has appeared in part in [8] and the *gap flow navigation* is inspired from our work in [9]. In this paper, we extend the tangential navigation by integrating the gap flow concept and by considering all obstacle points while computing the avoidance maneuver. Hence, the smoothness of the trajectories has been increased, and therefore a much more stable behavior is achieved. Furthermore, several experiments using our mobile robot, GETbot, in very dense and cluttered environments are provided. We also introduce a performance evaluation to quantitatively estimate the power of the proposed approach, and discuss and compare it with existing ND variants on the basis of the above mentioned limitations.

The remainder of the paper is structured as follows: After discussing the related work in Section 2, we present the reactive navigation method design in Section 3. In Section 4, we show and discuss the experimental results. Finally, Section 5 highlights our conclusions and future work.

2. Related work

In this section, we focus on the local reactive navigation techniques since global motion planning is out of the scope of this paper, and the reader is directed to [10] or [11] for an extended knowledge and taxonomy.

Early work in this topic includes the Artificial Potential Field (APF) approaches, initially proposed by Khatib [6]. The main idea stems from the gravitational force field principle. Within this concept, the robot is repelled from obstacles and attracted towards the goal by assuming that opposite forces are applied to the robot from the goal and surrounding obstacles. The resultant force determines the robot's subsequent direction and motion equations. Researchers have come up with a plethora of proposals to enhance this concept (e.g. [12–14]). Potential field techniques are considered to be fast and computationally efficient. Unfortunately, getting stuck in local minima and failure to find an oscillation-free motion in narrow passages are significant problems of these methods [7]. The Vector Field Histogram (VFH) [15] was then introduced mapping two-dimensional occupancy information into a one-dimensional histogram representation, which is then analyzed to detect potential open areas. Although this method produces smoother behavior and allows robots to travel at faster speeds without getting unstable [16], it, like the APF approach, can get trapped in local minima. Some works address the oscillation problem, such as [17], by using a modified Newton method, or [18], by employing a family of 2D smooth vector fields. Other works address the local minima problem, such as [19] by using a random walk if a local minimum is reached, or [20] by employing a special artificial potential function. While these techniques provide nice solutions to the APF drawbacks, they are either computationally expensive, based on strict assumptions, or not effective in complex environments [21].

The approaches defined in [22] and [23] use a very simple criterion to reach the goal. The appearance of obstacles in the vicinity of the robot pushes it to escape from the closest one by temporarily switching the desired goal location into a virtual one until the risk is passed. The concepts of APF and Tangential Escape (TE) are used for setting the new virtual goal location in [22] and [23], respectively. The latter approach generates smoother robot trajectories than the former one. Although such approaches are very simple to implement and result in faster reactions, they cease to function in slightly complex environments.

Other common techniques take the dynamic constraints of the robot into account and choose a steering command rather than a moving direction. The Curvature Velocity method (CVM) [24,25], and the Dynamic Window (DWA) approach [26,27] are the most popular ones. They work by adding constraints, coming from physical limitations and sensory data, to the velocity space, and choose the speed that satisfies all constraints and maximizes an objective function. While these techniques yield fast and smooth behavior, they may fail to drive the robot between close obstacles. Moreover, the robot can get stuck in local minima.

Several reactive methods are based on the concept of Velocity Obstacles (VO) [28–30] or Inevitable Collision States ICS [31,32]. These approaches consider the velocity of moving obstacles in determining the avoidance maneuver. VO-based approaches perform collision avoidance by identifying the set of robot's velocities that may cause collision at some future time, and select velocities outside of this set. ICS-based methods characterize all vehicle states that lead to collision at a later time in the future, and avoid these states while planning the robot's motion. A VO variant, the reciprocal velocity obstacle RVO [33,34], addresses the problem of cooperative collision avoidance. Although these techniques are applicable for static and dynamic obstacles, they assume known or predictable obstacle velocity. However, in real applications it is hard to predict the future of the scene [35]. Moreover, VO-based approaches require a careful determination of the time horizon. Otherwise, robots find the difficulty of passing through narrow spaces in dense and cluttered environments [36].

We close this section by an overview of the obstacle avoidance algorithms designed for dense and cluttered environments. The

Table 1
Overview of reactive obstacle avoidance navigation methods.

Approach	Environment	Strategy	Advantages	Disadvantages	Remarks
APF	Configuration space	Robot is repelled from obstacles and attracted towards the goal.	Fast, computationally efficient.	Local minima, oscillatory motion, point-like robot, disregards the kinodynamic constraints.	Robot shape and kinodynamic constraints can be considered following [38].
VFH	Certainty grid	Maps the environment into a 1D histogram and uses it to detect potential open areas.	Smoother and faster than APF.	Local minima, needs more processing time than APF, assumes a point-like robot, disregards the kinodynamic constraints.	A variant of VFH, VFH+ [39], assumes a circular-shaped robot and considers the kinematic constraints.
TE	Workspace	Drives the robot towards the goal, unless an obstacle gets close to it, in which case, moves tangent to the obstacle.	Simple to implement, computationally efficient.	Local minima, assumes a point-like robot, disregards the kinodynamic constraints.	The approach in [22] is similar to the TE, but avoids obstacles based on the extended impedance concept.
CVM, DWA	Velocity space	Adds constraints to the velocity space and then maximizes an objective function.	Smooth and fast behavior, considers the dynamic constraints (circular arcs).	Local minima, disregards the robot shape, fails to drive the robot between close obstacles.	The DWA has been extended in [27] to consider moving obstacles.
VO	Velocity space	Identifies the set of velocities that may cause collision, and selects velocities outside of this set.	Explicitly considers velocity of obstacles (within a predefined time horizon), considers the set of reachable velocities (dynamics).	Assumes known obstacle velocity, requires a careful selection of the time horizon, assumes a circular-shaped robot and obstacles.	Designed for moving obstacles. A variant of VO, RVO, is designed for cooperative collision avoidance.
ICS	State space	Characterizes all states that lead to collision in the future, and avoids these states.	Explicitly considers velocity of obstacles (infinite time horizon), considers dynamic constraints.	Requires a complete knowledge of the states of the environment, computationally expensive.	Designed for moving obstacles.
ND-based	Workspace	A divide and conquer strategy is used to identify the navigational situation, and then generates the corresponding motion law.	Avoids local trap situations, able to compute motion directions towards obstacles or far away from the goal, can drive the robot between closely spaced obstacles.	Oscillatory motion in narrow passages, deflection towards free areas, assumes a circular-shaped robot, does not consider the kinodynamic constraints.	Can drive the robot in cluttered and dense environments, robot shape and kinodynamic constraints have been considered in [38].

most popular one is the Nearness-Diagram (ND) [2] which uses a "divide and conquer" strategy to identify the navigational situation based on the data perceived by sensors, and then generates the corresponding motion law. ND identifies five different navigational laws, whereas the enhanced method named ND+ [37] adds another one. Afterwards, Minguez improved the behavior of ND+ in open spaces [3]. The Smooth Nearness-Diagram (SND) method [4] was then developed which proposes a single motion law to be applicable to all possible configurations of surrounding obstacles. In this regard, the robot's trajectory is based on all nearby obstacles, not just the closest two, and this leads to smoother paths. However, a problem of deadlock can appear in the SND in a narrow corridor if the difference in the number of threats on its sides is high. The Closest Gap (CG) approach proposed in our earlier work [5] overcame this drawback by considering the ratio of threats on both sides of the robot and by providing a stricter deviation against the nearest obstacles. Furthermore, a new algorithm for analyzing gaps was introduced that alleviates oscillations and reduces computational complexity. Experiments show that these approaches operate well in complex scenarios and avoid local traps. Navigation in such environments with faster speed, less oscillatory, and more stable behavior is the motivation of our work.

Table 1 provides a summary of the different approaches mentioned above, showing their advantages and disadvantages.

3. The reactive navigation method design

This section presents the Tangential Gap Flow Navigation (TGF) method for collision avoidance which fills in the role of a local planner that follows a perception–action process repeated at a high frequency rate. For simplicity, we assume a circular and holonomic mobile robot. However, an abstraction layer can be constructed to take into account the vehicle shape and non-holonomic constraints as has been addressed in [38]. Building and integrating this layer with the TGF is out of the scope of the paper and the reader is directed to [38] for more information. The TGF method works as follows: First, the sensory data is analyzed to determine the structure of obstacles surrounding the robot. Based on this analysis, the current situation is identified and its corresponding action is performed as described in Section 3.2. Mainly, these actions compute the instantaneous direction of motion which makes the best progress towards the goal whilst avoiding collisions. In Section 3.3, we describe how we set the motion commands that guide the robot towards the selected direction.

3.1. Assumptions and notations

The following assumptions and notations are used to explain our algorithm (see Fig. 1). Some of these notations have been used in [5]:

The robot and goal locations are denoted by $\mathbf{P}_r$ and $\mathbf{P}_g$, respectively. The robot is wrapped into a circle whose radius is denoted by R.

A *gap* is a potentially free path wide enough for the robot to move through [5]. Each gap has two sides, one is to the left of the other relative to the laser scanner frame. We call it a *left side* of the gap and the other is a *right side*.

The maximum range of the laser scanner is denoted by d_{max}. The list of scan points is denoted by S where a scan point is $\mathbf{p}_i^S$, and $i = 1, \ldots, n$ is the natural order of the range scans. 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.

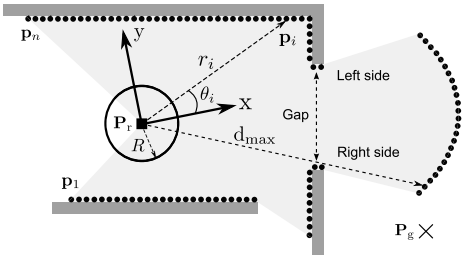

Fig. 1. Assumptions. The circle represents the robot and the dark gray regions are obstacles. The sensor data is shown by light gray color, where the small black circles represent the list of returned scan points S. In this figure, the S superscript has been removed for readability.

The positive x axis points to the front of the robot and the positive y axis is the normal on its left side. Angles are limited to the range $[-\pi, \pi[$. Let $\mathbb{S}^1$ be the unit circle attached to the robot's reference frame. For any two angles $\theta_1, \theta_2 \in \mathbb{S}^1$, the minimum angular distance between them is $\angle(\theta_1, \theta_2) = \min(\angle(\theta_1 \rightarrow \theta_2), \angle(\theta_1 \leftarrow \theta_2))$, where $\angle(\theta_1 \rightarrow \theta_2) = (\theta_1 - \theta_2) \mod 2\pi$ and $\angle(\theta_1 \leftarrow \theta_2) = (\theta_2 - \theta_1) \mod 2\pi$.

Let $a < b$, a saturation function is used to limit a value between a and b:

$$\Omega_{[a,b]}(x) = \begin{cases} a, & \text{if } x \leq a \\ x, & \text{if } a < x < b \\ b, & \text{if } x \geq b. \end{cases} \tag{1}$$

An angle θ may become greater than π or less than $-\pi$ during calculations. In order to map it into the range $[-\pi, \pi[$, a projection function is defined as:

$$\Pi(\theta) = ((\theta + \pi) \mod 2\pi) - \pi. \tag{2}$$

The Euclidean distance between two points $\mathbf{p}_a$, $\mathbf{p}_b$ is denoted by $\mathrm{d}(\mathbf{p}_a, \mathbf{p}_b)$.

3.2. Situations and associated actions

In order to achieve a safe navigation in a space occupied by obstacles, it is necessary to change the current robot's heading to avoid collisions. Hence, the goal location is temporarily rotated to another point until the risk is passed. The angle of rotation is calculated based on two criteria:

Criterion 1: Path to goal criterion. Two situations are considered here: Free-path and Dangerous-path. The current situation is determined as follows: first, the configuration space is constructed, then the line segment that connects the robot to the goal is checked. If it intersects any obstacle in the configuration space, the situation is a Dangerous-path. If no intersection occurs, it is a Free-path. In case of a Free-path situation, no action is required as shown in Fig. 2(a). But in case of a Dangerous-path, the goal location is rotated by an angle, named *gap rotation angle* (denoted by φ_{sg}), relative to the current robot's heading. The new goal location, which we refer to as a subgoal, causes the robot to move towards the navigable gap closest to the goal as shown in Fig. 2(b).

Criterion 2: Safety criterion. We consider two situations: High-safety and Low-safety. The current situation is identified based on checking whether a predefined security zone D_s around the robot contains obstacles or not. If so, the situation is a Low-safety. Otherwise, it is a High-safety. Only for a Low-safety situation the goal/subgoal position is rotated by an angle, named *collision avoidance rotation angle* (denoted by φ_{vg}). It is computed in such a way that the robot avoids the risk of collision with nearby obstacles while progressing towards the goal. The new location is referred to as a virtual-goal (see Fig. 2(c)). For a High-safety situation, no change to the actions associated with criterion 1 is required. Notice that the actions corresponding to criterion 1 must be performed before those corresponding to criterion 2.

At each control cycle, the instantaneous location of the goal is denoted by $\mathbf{P}_{\mathrm{cg}}$ and computed as:

$$\mathbf{P}_{\mathrm{cg}} = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix} \mathbf{P}_{\mathrm{g}} \tag{3}$$

where $\varphi = \varphi_{\mathrm{sg}} + \varphi_{\mathrm{vg}}$ is the total rotation angle.

First, we describe the procedure used to compute the gap rotation angle φ_{sg} in Section 3.2.1. Subsequently, in Section 3.2.2, we explain how the collision avoidance rotation angle φ_{vg} is determined based on the *tangential navigation* concept. Here, φ_{vg} is calculated by considering only the obstacle point closest to the robot. Later, in Section 3.2.3 the smoothness of the trajectory is improved by taking all obstacles into account and by integrating the *gap flow* concept.

3.2.1. Gap rotation angle

In order to identify the gap rotation angle φ_{sg}, we first analyze the range data to determine potential openings surrounding the robot. In [5] we have developed a novel method for such a purpose. A review of this method is introduced at first, and subsequently, a description of how we calculate φ_{sg} is presented.

Finding out gaps can be summarized in two steps: first, the list of all visible gaps is extracted, and then useless gaps are removed from this list. This is mainly based on detecting discontinuities between two successive range measurements (i, j), which can be of two types:

Edge discontinuity: Occurs when the difference between the depth measurements of scan numbers i and j exceeds $2R$, i.e. $\mathrm{d}(\mathbf{P}_{\mathrm{r}}, \mathbf{p}_j^S) - \mathrm{d}(\mathbf{P}_{\mathrm{r}}, \mathbf{p}_i^S) > 2R$.

Max-range discontinuity: Occurs if one of the two range measurements returns the maximum sensor range, i.e. $\mathrm{d}(\mathbf{P}_{\mathrm{r}}, \mathbf{p}_j^S) = \mathrm{d}_{\max} \wedge \mathrm{d}(\mathbf{P}_{\mathrm{r}}, \mathbf{p}_i^S) < \mathrm{d}_{\max}$.

Let $j > i$, if $r_j^S > r_i^S$ a *rising discontinuity* occurs at index i; else, it will be a *descending discontinuity* at index j. An edge discontinuity has a higher priority than a max-range. Next, we use these definitions to describe both steps.

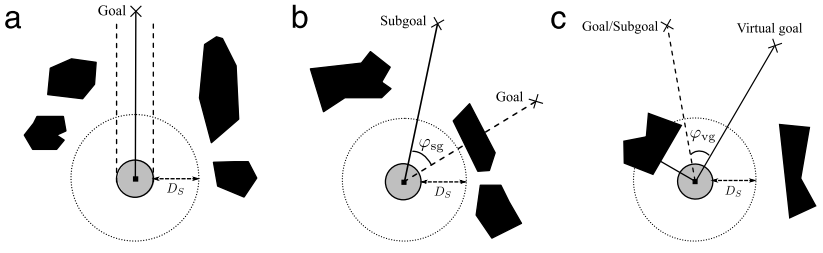

Fig. 2. Examples showing different situations and their corresponding actions. (a) A Free-path and High-safety situation. No action is performed. (b) A Dangerous-path and High-safety situation. The goal position is rotated so that the robot moves towards the gap closest to the goal (subgoal). (c) A Low-safety situation. The goal/subgoal location is temporarily rotated to a virtual goal to avoid the risk of collision.

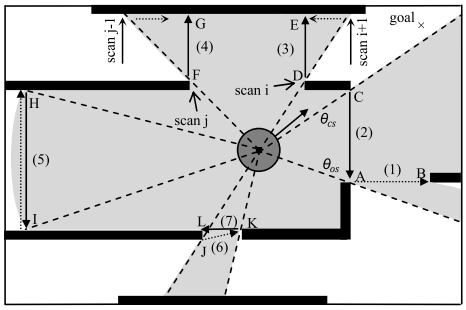

Fig. 3. Analyzing gaps by the CG method [5]. At first, gaps that are labeled 1, 3, 5, and 6 are detected by the *forward search* and gaps marked as 7, 4, and 2 are detected by the *backward search*. Then, gaps 1 and 6 shown by dashed arrows are eliminated. Finally, the gap closest to the goal (gap 2 in this figure) is chosen to navigate through.

Step 1: Searching for gaps is performed twice; *forward* and *backward*. The *forward search* is carried out by extracting rising discontinuities traveling from scan index 1 to n. The scan point at which a rising discontinuity occurs creates the first side of a gap (e.g. points D and H in Fig. 3). Let i be the index of this point, identifying the second side is based on the type of the discontinuity. For an edge discontinuity, it will be at index $j \geq i+1$ such that the distance between the obstacle point it returns and the first side is the minimum and the angular distance traveled must be less than π (e.g. Fig. 3, point E). For a max-range discontinuity, the second side occurs at index j which creates the first descending discontinuity, either type edge or max-range, coming after scan index i, see point I, in Fig. 3. The search process is resumed at scan index j producing gaps 1, 3, 5 and 6 in Fig. 3. In a similar way, the backward search is performed, but this time descending rather than rising discontinuities are extracted, while traveling from scan index n to 1. Gaps 7, 4 and 2 in Fig. 3 are extracted in the backward search.

Step 2: Once the list of all visible gaps is determined, every gap that falls within another gap is eliminated (e.g. gaps 1 and 6 in Fig. 3). Next, any gap having a width less than $2R$ is removed from the list (e.g. gap 7 in Fig. 3).

Among the assembled list of gaps, the one closest to the goal, named *closest gap*, is chosen (gap 2 in Fig. 3). It is determined by checking both sides of each gap and selecting the *navigable* one with the side making the minimum angular distance with the goal (for determining the navigability status of a gap, see [5]). The side of the closest gap closest to the goal is referred to as $\mathbf{p}_{cs}$ (point C in Fig. 3) and the other as $\mathbf{p}_{os}$ (point A in Fig. 3).

Computing φ_{sg} is based on the width of the closest gap and the location of the goal. If the goal falls within the closest gap sides, φ_{sg} is set to zero. Otherwise, φ_{sg} is computed in such a way that the robot moves towards the center of the closest gap if it is narrow. But for a wide closest gap, the robot navigates so that D_s is maintained between $\mathbf{p}_{cs}$ and the robot's boundary:

$$\varphi_{sg} = \begin{cases} \theta_{mid} - \theta_g, & \text{if } \angle(\theta_{cs}, \theta_{mid}) < \angle(\theta_{cs}, \theta_{scs}) \\ \theta_{scs} - \theta_g, & \text{otherwise} \end{cases} \tag{4}$$

where θ_{cs} is the angle towards $\mathbf{p}_{cs}$, θ_g the angle towards the goal, and θ_{mid} and θ_{scs} are defined as follows (originally from [4]):

$$\theta_{mid} = \begin{cases} \theta_{cs} - \dfrac{(\theta_{cs} - \theta_{os})}{2}, & \text{if } \theta_{cs} \text{ is a left side} \\ \theta_{cs} + \dfrac{(\theta_{os} - \theta_{cs})}{2}, & \text{otherwise} \end{cases} \tag{5}$$

$$\theta_{scs} = \begin{cases} \theta_{cs} - \arcsin\left(\dfrac{R + D_s}{r_{cs}}\right), & \text{if } \theta_{cs} \text{ is a left side} \\ \theta_{cs} + \arcsin\left(\dfrac{R + D_s}{r_{cs}}\right), & \text{otherwise} \end{cases} \tag{6}$$

where r_{cs} is the distance to $\mathbf{p}_{cs}$ and θ_{os} the angle towards $\mathbf{p}_{os}$.

3.2.2. Collision avoidance rotation angle

With the instantaneous goal (goal/subgoal) identified after checking criterion 1 situations, the TGF will consider rotating its location by φ_{vg} based on the configuration of nearby obstacles. From now on, the goal or the subgoal is called the target based on checking the situations concerning the path-to-goal criterion. The collision avoidance rotation angle φ_{vg} is determined so that the robot follows a path that is tangent to the closest obstacle boundary in the direction leading to the target:

$$\varphi_{vg} = \begin{cases} -\text{sgn}(\beta)\dfrac{\pi}{2} - \gamma, & \text{if } |\gamma| < \pi \wedge \text{sgn}(\alpha) \neq \text{sgn}(\beta) \\ -\text{sgn}(\beta)\dfrac{3\pi}{2} - \gamma, & \text{if } |\gamma| \geq \pi \wedge \text{sgn}(\alpha) \neq \text{sgn}(\beta) \\ -\text{sgn}(\beta)\dfrac{\pi}{2} - \gamma, & \text{if } |\beta| \geq |\alpha| \wedge \text{sgn}(\alpha) = \text{sgn}(\beta) \\ +\text{sgn}(\beta)\dfrac{\pi}{2} - \gamma, & \text{if } |\beta| < |\alpha| \wedge \text{sgn}(\alpha) = \text{sgn}(\beta) \end{cases} \tag{7}$$

where α is the angle towards the target, β the angle towards the obstacle point closest to the robot, and $\gamma = \alpha - \beta$.

Fig. 4 shows different situations and their associated φ_{vg} where the new target after rotating it by φ_{vg} is labeled Virtual Goal. It is clear that subgoals introduced by the TGF helps in selecting the correct direction for escaping. The TE method [23] is based on a similar concept. However, in some situations, the escaping angle proposed in [23] causes the robot to follow a path that is far away from the target. For example, in Fig. 4(b) and (d), the robot will be driven towards the direction visualized by the dashed arrows (see [8] for more details). The leaving condition is fulfilled if the angular difference between β and α exceeds $\frac{\pi}{2}$, i.e. the value of φ_{vg} is set to zero if $\angle(\alpha, \beta) > \frac{\pi}{2}$.

3.2.3. Trajectory smoothing

In Section 3.2.2, only one laser scan point (the closest one) has been considered to estimate φ_{vg}. This works fine if the environment is structured (e.g. polygonally shaped objects), which is not always the case in real applications. Moreover, the range measurements are error prone due to the uncertainty caused by sensors and the environmental changes. Hence, the location of the closest obstacle may change frequently over time causing undesirable motion behavior (e.g. an oscillatory behavior), particularly if the robot is moving at a relatively high speed. A similar yet more severe problem occurs when the robot travels in narrow corridors, in which the location of the closest obstacle changes from one side to the other. An example is shown in Fig. 5 where the robot travels through the narrow passage between the sides labeled A and B (assuming that we have a Dangerous-path and Low-safety situation, and both sides are falling within D_S). $\mathbf{P}_t$ represents the target location (subgoal here). When the robot is at the location labeled 1 (Fig. 5(a)), the closest obstacle point is on side A. Hence, $\mathbf{P}_t$ is rotated by φ_{vg}, and the robot moves parallel to side A obstacles. As soon as the robot reaches location 3 (Fig. 5(b)), the closest obstacle point is on side B, and the new φ_{vg} will head the robot parallel to side B obstacles. At point 4, the closest obstacle point is again on side A switching the target location as shown in Fig. 5(c). Whenever the angular width between side B obstacles and $\mathbf{P}_t$ gets approximately $\frac{\pi}{2}$ (Fig. 5(d)), the leaving condition is satisfied and the robot moves directly towards $\mathbf{P}_t$. These sudden changes excite the robot into an oscillatory motion. Notice that if the gap is wide enough, the leaving condition is satisfied before the robot gets closer to side B, and thus, no oscillation occurs.

To overcome such situations, we adapt φ_{vg} proposed in Section 3.2.2 by taking all nearby obstacle points into account, not just the closest one. Furthermore, the free space between obstacles on both sides of the target is taken into consideration

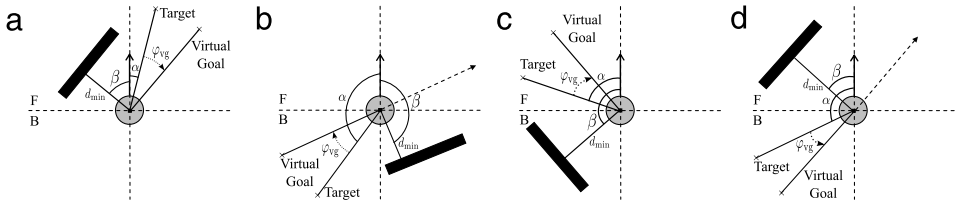

Fig. 4. Different situations and their associated collision avoidance rotation angles. (a, b) $\text{sgn}(\alpha) \neq \text{sgn}(\beta)$ where $|\gamma| < \pi$ for (a) and $|\gamma| \geq \pi$ for (b). (c, d): $\text{sgn}(\alpha) = \text{sgn}(\beta)$ where $|\beta| \geq |\alpha|$ for (c) $|\beta| < |\alpha|$ for (d).

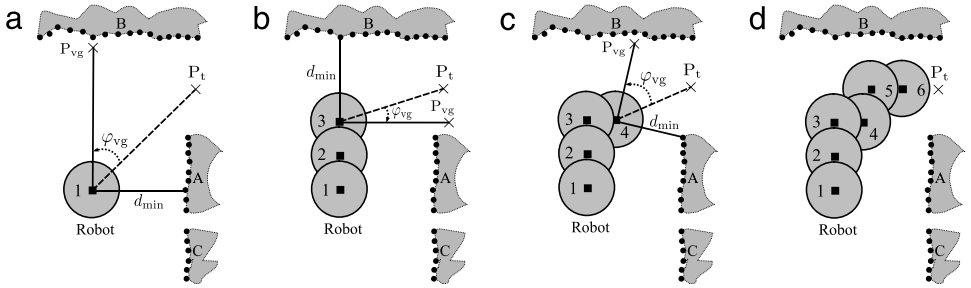

Fig. 5. Oscillations occurring in narrow passages using the collision avoidance rotation angle proposed in Eq. (7). (a) The closest obstacle point is on side *A*. Therefore, $\mathbf{P}_t$ is rotated by φ_{vg}, and the robot moves parallel to side A obstacles accordingly. (b) The closest obstacle point is on side B, and the new φ_{vg} will head the robot parallel to side B obstacles. (c) The closest obstacle point is again on side A and the robot moves parallel to it. (d) The leaving condition is satisfied and the robot moves towards the target directly.

when setting φ_{vg}. As a consequence, a much more stable and smoother behavior is achieved. The improved rotation angle is referred to as *a tangential gap flow rotation angle* (TGF rotation angle) and denoted by Ψ_{vg}.

The foundation of Ψ_{vg} is the measurement of the risk (threat) posed by each of the obstacle points around the robot. In a nutshell, Ψ_{vg} is calculated as follows: first, we identify the set of obstacle points considered as threats while traveling towards the target. Then, we calculate the rotation angle caused by each threat ψ_i considering the free space between obstacles on both sides of the target. Finally, the weighted average rotation angle is computed based on the relative proximity of each threat.

3.2.3.1. Determining threats. The procedure of identifying threats is described in the following. In a first step, we translate the target to a safe point within the closest gap in case that the current situation concerning criterion 1 is a Dangerous-path (see Fig. 6). The distance to the new target location ρ is:

$$\rho = \begin{cases} d\left(\mathbf{P}_r, \mathbf{P}_g\right), & \text{if } \varphi_{sg} = 0 \\ \dfrac{r_{cs}\ \sin\left(\zeta\right)}{\sin\left(\pi - (\zeta + \eta)\right)}, & \text{otherwise} \end{cases} \tag{8}$$

where r_{cs} is the distance to $\mathbf{p}_{cs}$, and ζ and η are given by:

$$\zeta = \arccos\left(\frac{w^2 + r_{cs}^2 - r_{os}^2}{2 w r_{cs}}\right) \tag{9}$$

$$\eta = \begin{cases} \angle\left(\alpha \leftarrow \theta_{cs}\right), & \text{if } \theta_{cs} \text{ is a left side} \\ \angle\left(\alpha \rightarrow \theta_{cs}\right), & \text{otherwise} \end{cases} \tag{10}$$

where r_{os} is the distance to the closest gap side farther from the goal, w the Euclidean distance between both sides, and α the angle towards the target.

In a second step, we create a reference frame, named *robot–target coordinate system*, by rotating the robot's frame to head towards the target (rotation by α). In the remaining part of this section, we denote the Cartesian coordinates of $\mathbf{p}_i^S$, relative to

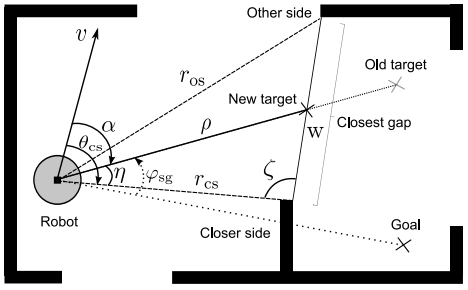

Fig. 6. Translating the target in a safe area within the closest gap if the current situation concerning criterion 1 is a Dangerous-path.

the robot–target coordinate system, by $(T(x_i^S),\ T(y_i^S))$, i.e.:

$$T(x_i^S) = x_i^S \cos(-\alpha) - y_i^S \sin(-\alpha) \tag{11}$$

$$T(y_i^S) = x_i^S \sin(-\alpha) + y_i^S \cos(-\alpha). \tag{12}$$

Let M be the set of obstacle points falling within D_s. Eliminate from M those points that are located behind the robot and those points that are far away from the current planned target. Notice that the obstacle points removed do not pose a danger of collision. We denote the resultant set of threats by N:

$$N = \left\{\mathbf{p}_i^S \in S : d(\mathbf{P}_r, \mathbf{p}_i^S) < D_s,\ 0 \leq T(x_i^S) \leq \rho\right\}. \tag{13}$$

The robot–target coordinate system divides this set into two subsets; one consists of threats located to the right of the x-axis (right side) and the other contains threats located to the left (left side). The former is called a *right-subset* (denoted by N_R), and the latter a *left-subset* (denoted by N_L):

$$N_R = \left\{\mathbf{p}_i^S \in N : T(y_i^S) < 0\right\} \tag{14}$$

$$N_L = \left\{\mathbf{p}_i^S \in N : T(y_i^S) > 0\right\}. \tag{15}$$

Among the obstacle points contained in each subset, we only consider those points having an absolute y-coordinate less than or equal to the one closest to the robot. In this regard, obstacle points

falling within U-shaped obstacles are excluded, and thus irrational deflections towards free areas are avoided (this results in shorter and safer trajectories). Let $\mathbf{p}_c^{N_R} \in N_R$ and $\mathbf{p}_c^{N_L} \in N_L$ be the obstacle points closest to the robot boundary in each subset, i.e.:

$$\mathbf{p}_c^{N_R} = \underset{\mathbf{p}_j^S}{\operatorname{argmin}} \|\mathbf{p}_j^S - \mathbf{P}_r\|, \quad \mathbf{p}_j^S \in N_R \tag{16}$$

$$\mathbf{p}_c^{N_L} = \underset{\mathbf{p}_j^S}{\operatorname{argmin}} \|\mathbf{p}_j^S - \mathbf{P}_r\|, \quad \mathbf{p}_j^S \in N_L. \tag{17}$$

The modified N_R and N_L are denoted by N_R' and N_L' and defined as follows:

$$N_R' = N_R \setminus \left\{\mathbf{p}_i^S \in N_R : \left|T(y_i^S)\right| > \left|T(y_c^{N_R})\right|\right\} \tag{18}$$

$$N_L' = N_L \setminus \left\{\mathbf{p}_i^S \in N_L : \left|T(y_i^S)\right| > \left|T(y_c^{N_L})\right|\right\} \tag{19}$$

where $y_c^{N_R}$ and $y_c^{N_L}$ are the y-coordinates of $\mathbf{p}_c^{N_R}$ and $\mathbf{p}_c^{N_L}$, respectively.

3.2.3.2. Tangential gap flow rotation angle. Each threat $\mathbf{p}_i^S$ contained in any of the two subsets (sides) causes a rotation angle ψ_i to the target position. Setting ψ_i is based on the locations of the target, $\mathbf{p}_i^S$, and the closest to $\mathbf{p}_i^S$ in the other subset. Let H represent the subset containing threat $\mathbf{p}_i^S$ and H^* the other subset. We denote threat $\mathbf{p}_i^S$ by $\mathbf{p}_i^H$ and the threat closest to $\mathbf{p}_i^S$ and falling in H^* by $\mathbf{p}_c^{H^*}$:

$$\mathbf{p}_c^{H^*} = \underset{\mathbf{p}_j^S}{\operatorname{argmin}} \|\mathbf{p}_j^S - \mathbf{p}_i^H\|, \quad \mathbf{p}_j^S \in H^*. \tag{20}$$

Before explaining how ψ_i is calculated, we will now define an angle called *gap flow angle* that guarantees a smooth and safe avoidance of $\mathbf{p}_i^H$. Let θ_c be the angle towards the center point between $\mathbf{p}_i^H$ and $\mathbf{p}_c^{H^*}$:

$$\theta_c = \operatorname{atan2}\left(\frac{y_i^H + y_c^{H^*}}{2}, \frac{x_i^H + x_c^{H^*}}{2}\right) \tag{21}$$

where (x_i^H, y_i^H) and $(x_c^{H^*}, y_c^{H^*})$ are the Cartesian coordinates of $\mathbf{p}_i^H$ and $\mathbf{p}_c^{H^*}$.

The gap flow angle Δ_i^H of threat $\mathbf{p}_i^H$ is defined as follows:

$$\Delta_i^H = \arccos\left(\frac{1}{2} \cdot \frac{d^2 + (r_i^H)^2 - {d_s}^2}{d\, r_i^H}\right) \tag{22}$$

where r_i^H is the distance to $\mathbf{p}_i^H$, and d and d_s are given by:

$$d_s = \begin{cases} \min\left(0.5\|\mathbf{p}_i^H - \mathbf{p}_c^{H^*}\|, D_{\max}\right), & \text{if } r_i^H \le r_c^{H^*} \\ \min\left(\left| r_i^H \sin\left(\angle\left(\theta_i^H, \theta_c\right)\right)\right|, D_{\max}\right), & \text{otherwise} \end{cases} \tag{23}$$

$$d = \sqrt{d_s^2 + \left(r_i^H\right)^2 - 2\, d_s \cdot r_i^H \cdot \cos\left(\frac{\pi}{2} - \angle\left(\theta_i^H, \theta_c\right)\right)} \tag{24}$$

where $r_c^{H^*}$ is the distance to $\mathbf{p}_c^{H^*}$, θ_i^H the angle towards $\mathbf{p}_i^H$, and $D_{\max}$ a parameter used to limit the gap flow angle in case that the clearance between $\mathbf{p}_i^H$ and $\mathbf{p}_c^{H^*}$ is big enough. We set $D_{\max} = 2R$ in our experiments. Δ_i^H defines the angular distance between $\mathbf{p}_i^H$ and the desired avoidance trajectory associated with $\mathbf{p}_i^H$. It is set in such a way that the robot smoothly and safely navigates towards the free area in between $\mathbf{p}_i^H$ and $\mathbf{p}_c^{H^*}$. If $\mathbf{p}_i^H$ is closer to the robot than $\mathbf{p}_c^{H^*}$, Δ_i^H heads the robot so that $\frac{1}{2}\|\mathbf{p}_i^H - \mathbf{p}_c^{H^*}\|$ is maintained between the robot and $\mathbf{p}_i^H$. For a large value of $\frac{1}{2}\|\mathbf{p}_i^H - \mathbf{p}_c^{H^*}\|$, it is preferable to limit the maintained distance to $D_{\max}$. But if $\mathbf{p}_c^{H^*}$ is closer to the robot than $\mathbf{p}_i^H$, Δ_i^H will point the robot towards the center point between $\mathbf{p}_i^H$ and $\mathbf{p}_c^{H^*}$. Analogously, if this will bring the robot much away from $\mathbf{p}_i^H$, we cap the distance between them to $D_{\max}$. Having in mind all Δ_i^N on both sides, the situation will be similar to creating a vector flow field between all $\mathbf{p}_i^H$ and $\mathbf{p}_c^{H^*}$ where the direction of flow is towards the closest gap (goal in case of a Free-path situation) and a suitable clearance to obstacles is preserved. That is why we call Δ_i^H the *gap flow* angle.

With this, we can define ψ_i applied to the target location to avoid $\mathbf{p}_i^H$ as:

$$\psi_i = \Gamma\left[-\lambda + \Omega_{[0,\lambda]}\left(\angle\left(\alpha, \theta_i^H\right)\right)\right] \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right[\tag{25}$$

where Γ and λ are given by:

$$\Gamma = \begin{cases} +\operatorname{sgn}\left(\theta_i^H - \alpha\right), & \text{if } \left|\theta_i^H - \alpha\right| \le \pi \\ -\operatorname{sgn}\left(\theta_i^H - \alpha\right), & \text{otherwise} \end{cases} \tag{26}$$

$$\lambda = \begin{cases} \dfrac{\pi}{2}, & \text{if } H^* = \phi \\ \Delta_i^H, & \text{otherwise.} \end{cases} \tag{27}$$

We use Γ in Eq. (25) to head the robot in the direction leading to the target instead of using the 4 cases in Eq. (7). If H^* is empty, the result of Eq. (25) is equivalent to Eq. (7), except that the leaving condition is included in the equation (*tangential navigation*). This is done using Ω which caps ψ_i at 0 when the angular difference between the target and $\mathbf{p}_i^H$ gets greater than $\frac{\pi}{2}$. However, if H^* is not empty, Eq. (25) heads the robot towards the free space between $\mathbf{p}_i^H$ and $\mathbf{p}_c^{H^*}$ using the gap flow angle (*gap flow navigation*).

In order to reflect the relative importance of each ψ_i, it is weighted based on the proximity of $\mathbf{p}_i^H$ to the robot's boundary. We use the following measure:

$$w_i = \left(\Omega_{[0,1]} \frac{\left(D_s - r_i^H\right)}{\left(D_s - r_c^{N'}\right)}\right)^2 \tag{28}$$

where $r_c^{N'}$ is the distance to the threat closest to the robot among those falling in N_R' and N_L' (i.e. $r_c^{N'} = \min(r_c^{N_R'}, r_c^{N_L'})$). The saturation operator Ω caps w_i at 0 when $\mathbf{p}_i^H$ is outside D_s and at 1 if $r_i^H = r_c^{N'}$. The value of w_i increases as the robot gets closer to the robot's boundary.

The weighted average rotation angle is then calculated for each subset separately. In this regard, the resultant TGF rotation angle Ψ_{vg} will not be influenced by the number of obstacle points on each side. Consequently, we avoid the problem of getting close to the side having a small number of obstacles compared to the other (for more information about this problem the reader is directed to [5]). Let T_L and T_R be the total number of threats contained in the left and right subsets, respectively. The weighted average rotation angle caused by threats falling in the left subset (left-side rotation angle) is then defined by:

$$\Psi_L = \frac{1}{W_L} \sum_{i=1}^{T_L} w_i \psi_i \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right[, \quad \mathbf{p}_i^S \in N_L' \tag{29}$$

where W_L is the total weight associated with threats falling in N_L', i.e.:

$$W_L = \sum_{i=1}^{T_L} w_i, \quad \mathbf{p}_i^S \in N_L'. \tag{30}$$

Analogously, we find the right-side rotation angle Ψ_R. Both left and right side rotation angles are weighted to reflect their relative importance. The weight associated with Ψ_L is defined as follows:

$$W(\Psi_L) = \underbrace{w_{\max}^L}_{\text{left-term}} \cdot \underbrace{\left(1 - \frac{\Psi_{\max} - |\Psi_L|}{\Psi_{\max}}\right)}_{\text{right-term}} \tag{31}$$

where w^{L}_{max} is the maximum weight assigned to threats located on the left side (the weight assigned to $\mathbf{p}_c^{N'_L}$), and Ψ_{max} the larger absolute value among the two side rotation angles ($\Psi_{max} = \max(|\Psi_R|, |\Psi_L|)$). As we can see, the value of $W(\Psi_L)$ is based on two factors: first (left-term), the proximity of $\mathbf{p}_c^{N'_L}$ to the robot boundary. We only consider $\mathbf{p}_c^{N'_L}$, among threats falling on the left side, since it poses the highest risk. Second (right-term), the relative difference between Ψ_{max} and $|\Psi_L|$. If $|\Psi_L|$ is higher than $|\Psi_R|$, the value of the right-term in Eq. (31) is 1. If not, the closer the value of $|\Psi_L|$ to $|\Psi_R|$, the higher the right-term will be. By this means, the side exerting a higher rotation (higher risk) will be dominant. This results in a smooth variation of Ψ_{vg} if the larger (dominant) avoidance angle changes between being Ψ_L or Ψ_R. The weight associated with Ψ_R ($W(\Psi_R)$) is calculated in the same manner.

With this we can now define the tangential gap flow rotation angle Ψ_{vg} as the weighted average of Ψ_R and Ψ_L:

$$\Psi_{vg} = \frac{W(\Psi_R)\Psi_R + W(\Psi_L)\Psi_L}{W(\Psi_R) + W(\Psi_L)} \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right[\tag{32}$$

3.3. Determining motion commands

In this section, we describe how to compute the motion commands that drive a mobile robot towards a given goal location ($\mathbf{P}_{cg}$ in our case). The robot considered here is a unicycle type whose control inputs are the translational and rotational velocities (v and w). The kinematic equations of such a robot, in polar coordinates, are defined as follows [40,41]:

$$\dot{\rho} = -v\cos(\delta) \tag{33}$$

$$\dot{\delta} = -w + v\frac{\sin(\delta)}{\rho} \tag{34}$$

$$\dot{\vartheta} = -v\frac{\sin(\delta)}{\rho} \tag{35}$$

where δ is the angle towards $\mathbf{P}_{cg}$ and ϑ the angle between the x-axis of the world coordinate system and the line to the goal.

In [8], we have developed a Lyapunov-based final pose controller to solve this problem. The control inputs are given by:

$$v = k_b v_{limit}\cos(\delta) \tag{36}$$

$$w = \Omega_{[-w_{max}, w_{max}]}\left(k_m\delta + \frac{v\sin(\delta)}{\rho}\right), \quad k_m > 0. \tag{37}$$

The saturation operator (Ω) in Eq. (37) is used to cap the value of w between the maximum negative and positive angular velocity ($-w_{max}, w_{max}$). The value of k_b in Eq. (36) is set to $\tanh(\rho)$ in case of a Free-path situation, and to 1 otherwise. By this means, a smooth breaking is guaranteed when the robot approaches the goal. The value of k_m in Eq. (37) is used to limit the maximum value of w, we set $k_m = \frac{2w_{max}}{\pi}$ in our experiments. The speed of the robot is controlled by specifying v_{limit} to maintain safety near close obstacles [5]:

$$v_{limit} = \left(\sqrt{1 - \Omega_{[0,1]}\left(\frac{D_{vs} - d_{min}}{D_{vs}}\right)}\right)\cdot v_{max} \tag{38}$$

where d_{min} is the distance to the obstacle point closest to the robot boundary, v_{max} the maximum translational speed of the robot, and D_{vs} a parameter, called velocity safe distance in [5]. The value of D_{vs} determines how much the speed is limited; increasing it results in a more conservative behavior.

4. Experimental results

Several experiments were carried out to verify that the TGF method complies with the goal of this work: *To safely drive a mobile robot in complex and cluttered environments with faster speed, smoother trajectory, shorter path, and more stable behavior, as compared with the ND navigation methods*. Before presenting these experiments in Section 4.2, we first introduce our experimental setup in Section 4.1. Subsequently, in Section 4.3, we describe the performance measures used to assess the behavior of the TGF method. Finally, a discussion and comparison with the ND-based navigation variants is presented in Section 4.4.

4.1. Experimental setup

We tested the TGF method on our rescue mobile robot GETbot, a skid-steering Pioneer 3-AT equipped with an on-board computer having a 2.6 GHz Intel core i5-3320M CPU and two laser rangefinders. The first laser scanner is a Hokuyo UTM-30LX attached to the front of the robot and the other is a Hokuyo URG-04LX located in the back. The former has a horizontal field of view of 270° and an angular resolution of 0.25° with a maximum range of 30 m, while the other has a horizontal field of view of 240° and an angular resolution of 0.35° with a maximum range of 5.6 m. The mobile robot has a rectangular shape (0.52 × 0.48 m) and non-holonomic constraints. The maximum linear speed is 0.7 m/s and the maximum rotational one is 2.4 rad/s. To get a 360° field of view, the laser data returned by both rangefinders are merged creating a virtual laser scanner with a coordinate system aligned to the robot's frame.

4.2. Experiments

In this section, we outline five experiments carried out in highly cluttered environments where none of the obstacle locations were known to the robot in advance. For the sake of comparison, these experiments were carried out using the ND+ [2], SND [4], and CG [5] nearness-diagram navigation methods, in addition to the proposed TGF approach. We limited the maximum linear and angular velocities while carrying out all experiments ((0.4 m/s, 0.8 rad/s) in experiment 1 and (0.5 m/s, 1.0 rad/s) in experiments 2–5). This is due to the limited capability of the ND variants to safely drive the robot at higher speeds (see discussion in Section 4.4). The safe distance D_s was set to 1 m in all experiments, while the velocity safe distance D_{vs} was set to 0.9 m in the CG and TGF methods. The weight strength k in the CG method was set to 0.9, 0.5, 0.6, 0.8, 0.8 in experiments 1–5, respectively. Notice that we tried several parameter settings before selecting the most suitable values for each experiment. In order to prevent cyclic loops, higher priority was given to gaps located in front of the robot than rear ones. All techniques were implemented using the Robot Operating System (ROS). For visualization purposes, we created maps of the environment using an open source SLAM algorithm available in ROS [42].

The ND variants control the robot's movement using motion commands (named *ND motion commands*) different from those proposed in Section 3.3 (*TGF motion commands*). In order to have a fair comparison, we carried out all experiments using the ND motion commands. The differences in behavior between the ND and TGF motion commands are then discussed in Section 4.4.

Experiment 1: In this experiment, the robot was supposed to navigate an obstacle course created by randomly distributed boxes as shown in Fig. 7(a). The trajectories followed by ND+, SND, CG, and TGF methods are shown in Fig. 7(b), (c), (d), and (e), respectively. The red rectangular boxes, plotted at regular intervals on the map, represent the progress of the robot. The

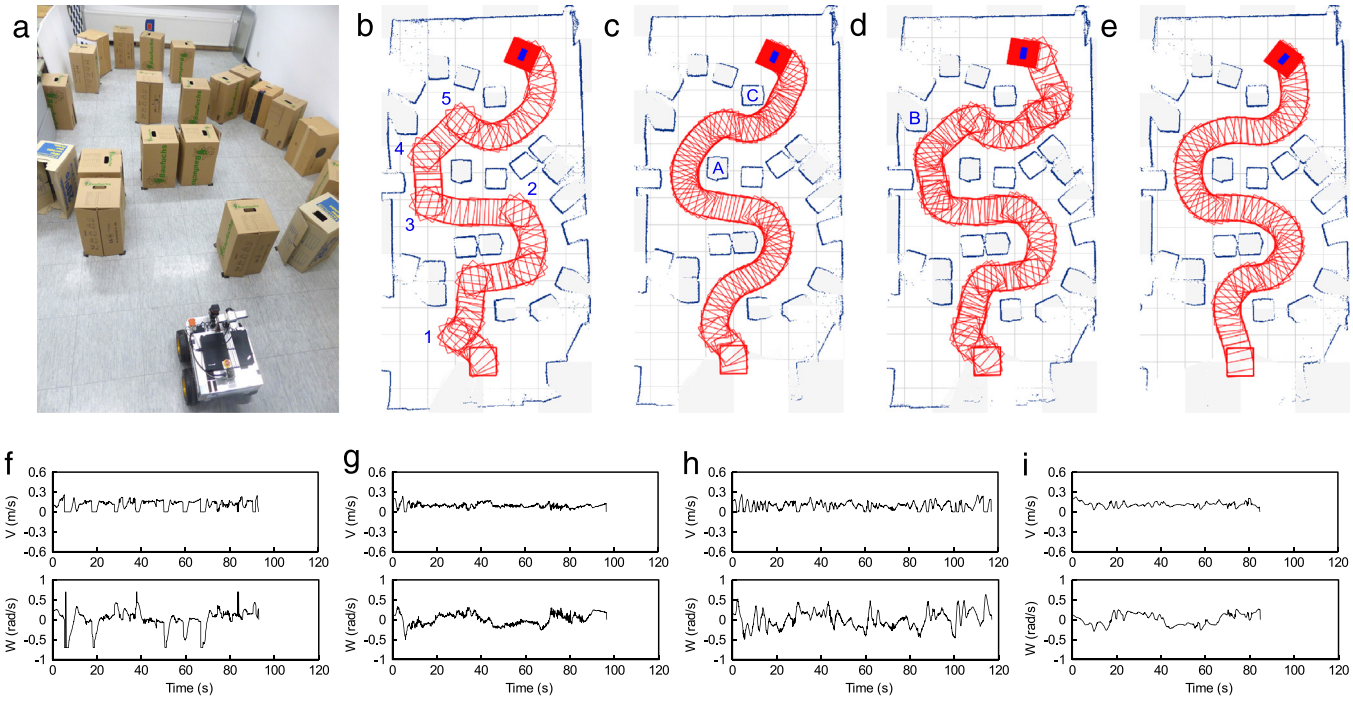

Fig. 7. Experiment 1. (a) Environment setup. (b–e) Trajectories followed by (b) ND+, (c) SND, (d) CG and (e) TGF. (f–i) Velocity profile for (f) ND+, (g) SND, (h) CG and (i) TGF. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

density of these boxes indicates the relative speeds at different locations. One can notice that sudden changes in the steering angle occurred along the path taken by the ND+ method causing sharp turns (e.g. points labeled 1–5 in Fig. 7(b)). Using the SND and CG methods, the robot navigated close to obstacles while passing narrow passages (e.g. see the trajectory near obstacles A–C in Fig. 7(c) and (d)). It is obvious from Fig. 7(e) that the robot safely and smoothly traversed the route using the TGF method. We recorded the linear and angular velocities over the course of the experiment. Fig. 7(f), (g), (h), and (i) show these velocities plotted versus time for the ND+, SND, CG, and TGF methods, respectively.

Experiment 2: In this experiment, a mixture of relatively thin and thick obstacles was created with a large U-shape structure (see Fig. 8(a)). The route chosen included very narrow openings (less than 10 cm at both sides). Applying the CG algorithm, the robot got close to obstacles (e.g. obstacles A–D in Fig. 8(d)). Moreover, the robot touched the obstacle labeled F running the CG and ND+ algorithms (see Fig. 8(b) and (d)). Using the SND technique, the robot did not complete the course after hitting obstacle D and finally pushing over the thin obstacles of the narrow passage created by obstacles A and E (Fig. 8(c)). Sharp changes in the steering angle can be noticed from the trajectories followed by the ND variants (e.g. points 1–3 in Fig. 8(b)–(d)). The TGF method drove the robot faster and smoother than the ND variants. This can be interpreted from the density of the red tails left behind on the map at regular intervals (see Fig. 8(e)) and from plotting the recorded velocities versus time (Fig. 8(f)–(i)).

Experiment 3: The scenario chosen for this experiment was a challenging one due to three main difficulties (see Fig. 9(a)). The first one was the highly reduced room available to maneuver through the complete mission (e.g. passages P2, P3, and P4). The second one was the narrow openings a long the traversed route, where the robot did not fit in. The existence of narrow passages with a large difference in the number of threats on their sides was the third difficulty (e.g. passages P1 and P2). The robot was able to traverse the route and reach the goal by applying ND+ and TGF methods (Fig. 9(b), (e)). However, while entering the first opening (labeled 1 in Fig. 9(b)) ND+ was unstable for a while. Moreover, the motion was oscillatory and the robot touched the chair labeled A in Fig. 9(b). Using the SND and CG techniques, the robot pushed over the thin obstacles of the first opening as shown in Fig. 9(c) and (d). Fig. 9(f)–(i) show the recorded motion commands against the time elapsed for all methods.

Experiment 4: Fig. 10(a) and (d) show the environmental setup of the fourth experiment. The robot should pass two obstacle courses to reach the goal. The first one is shown in Fig. 10(a) (between the starting location and the door) and the other is shown in Fig. 10(d) (between the door and the goal location). The main difficulty in this scenario was the mixture of relatively narrow and wide openings (e.g. passages P1 and P2, P3 and P4). In this regard, the robot was suddenly changing the velocity (decelerating when moving from wide to narrow areas and vice versa). Moreover, several U-shaped obstacles should be avoided to reach the goal (e.g. see Fig. 10(d)). The robot successfully passed the course by using all methods. Fig. 10(b), (c), (e), and (f) show the paths followed by ND+, SND, CG, and TGF methods, respectively. The ND variants, like the previous experiments, suffered from suddenly changing the steering angle resulted in an oscillatory motion (e.g. see the trajectory at the points labeled 1–8 in Fig. 10(b), (c), and (e)). This is obvious in the velocity profiles of these methods (see the large spikes in Fig. 10(g)–(i)). Furthermore, the robot was alternating between right and left turns while moving through passage P5 when we applied the ND+ algorithm (see Fig. 10(b)). Using CG, the robot pushed over the obstacle marked as D and moved very close to obstacle E. The behavior was worse running the SND algorithm since the robot navigated close to obstacles at different locations (e.g. obstacles A–C) and pushed over obstacles D, E, and F. The increased speed and smoothness of the trajectory generated by the TGF approach is clearly demonstrated in Fig. 10(f) and (j).

Experiment 5: In this experiment we tested the performance of the TGF in the presence of dynamic obstacles. Fig. 11(a) shows the initial situation where the corridor between the two boxes was free. At that time, the robot detected the navigable gap marked as G1 and moved directly towards it. As soon as the robot crossed the line labeled L, 3 boxes were pushed inside the corridor. At

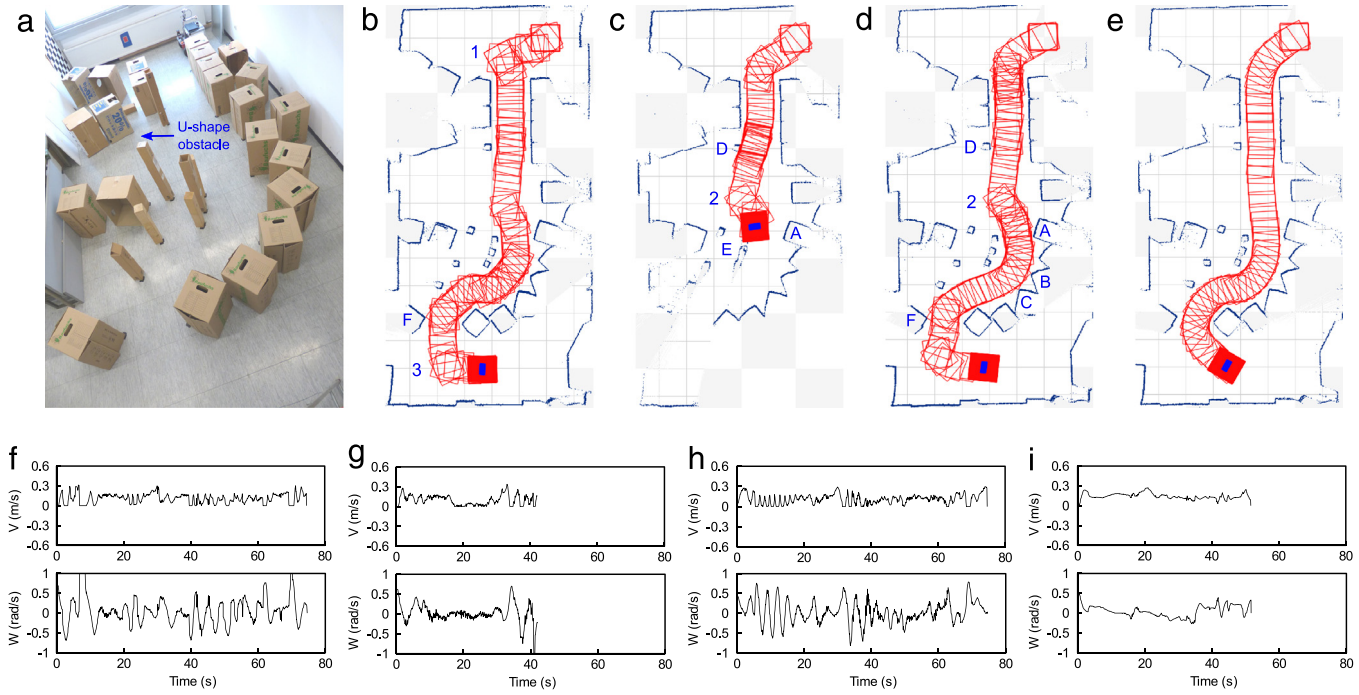

Fig. 8. Experiment 2. (a) Environment setup. (b–e) Trajectories followed by (b) ND+, (c) SND, (d) CG and (e) TGF. (f–i) Velocity profile for (f) ND+, (g) SND, (h) CG and (i) TGF. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

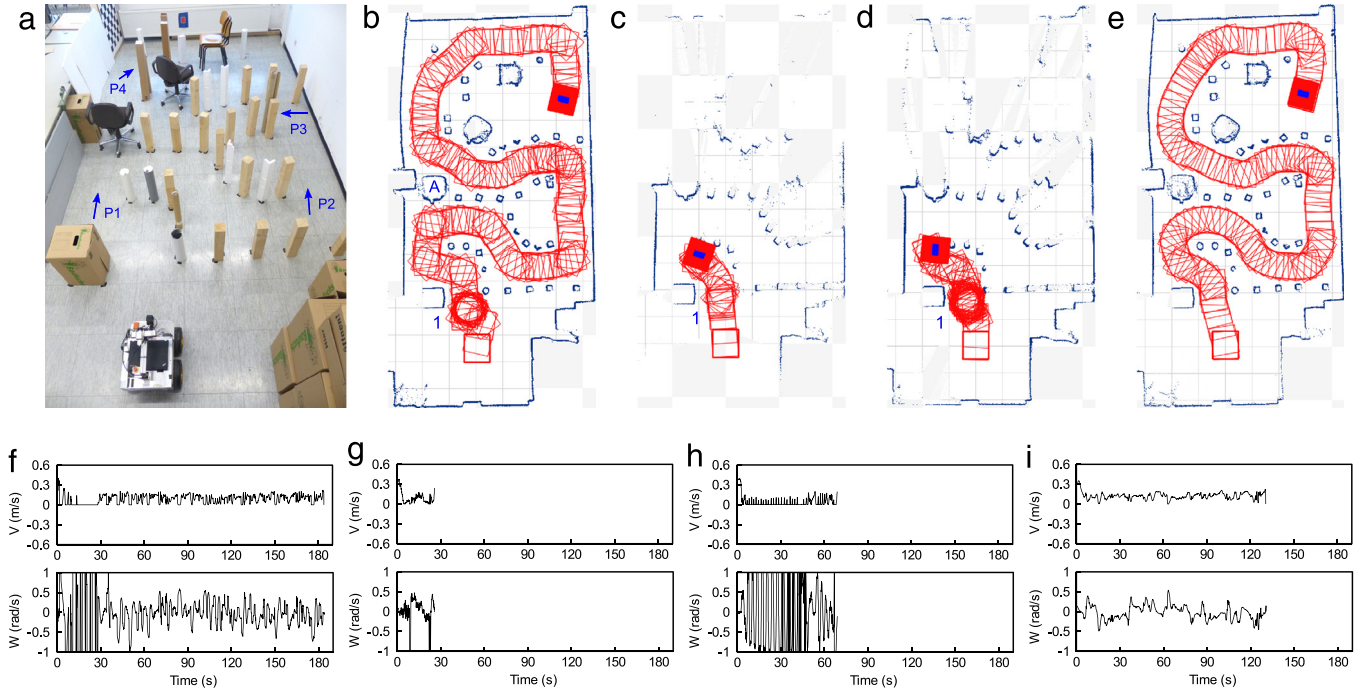

Fig. 9. Experiment 3. (a) Environment setup. (b–e) Trajectories followed by (b) ND+, (c) SND, (d) CG and (e) TGF. (f–i) Velocity profile for (f) ND+, (g) SND, (h) CG and (i) TGF.

that moment, the distance between line L and the boxes was 50 cm approximately (see Fig. 11(b)). The robot detected that gap G1 was closed and a new navigable gap to the right-hand was created (located between the boxes and the cupboard, and denoted by G2 in Fig. 11(b)). At first, the vehicle did a sharp turn to avoid the boxes. Then, it smoothly moved towards gap G2 and proceeded towards the goal. Fig. 11(f) and (j) show the trajectory followed by the TGF and the velocity profile, respectively. We tested the ND variants nearly under the same conditions. Notice that it is unfair to directly compare the algorithms since it was impossible to push the boxes exactly at the same speed, time, and direction. Moreover, each method was reaching line L with different speed and orientation. The speed of the robot was 0.11 m/s, 0.045 m/s, 0.033 m/s, and 0.21 m/s using ND+, SND, CG, and TGF, respectively. However, just to give the reader a feeling of how each method behaves, we provide the trajectories followed by ND+, SND, and CG methods in Fig. 11(c)–(e), and the velocity profiles in Fig. 11(g)–(i), respectively. Despite the fact that the robot reached line L at a higher speed using the TGF method, it reacted on time and smoothly avoided the boxes without collision.

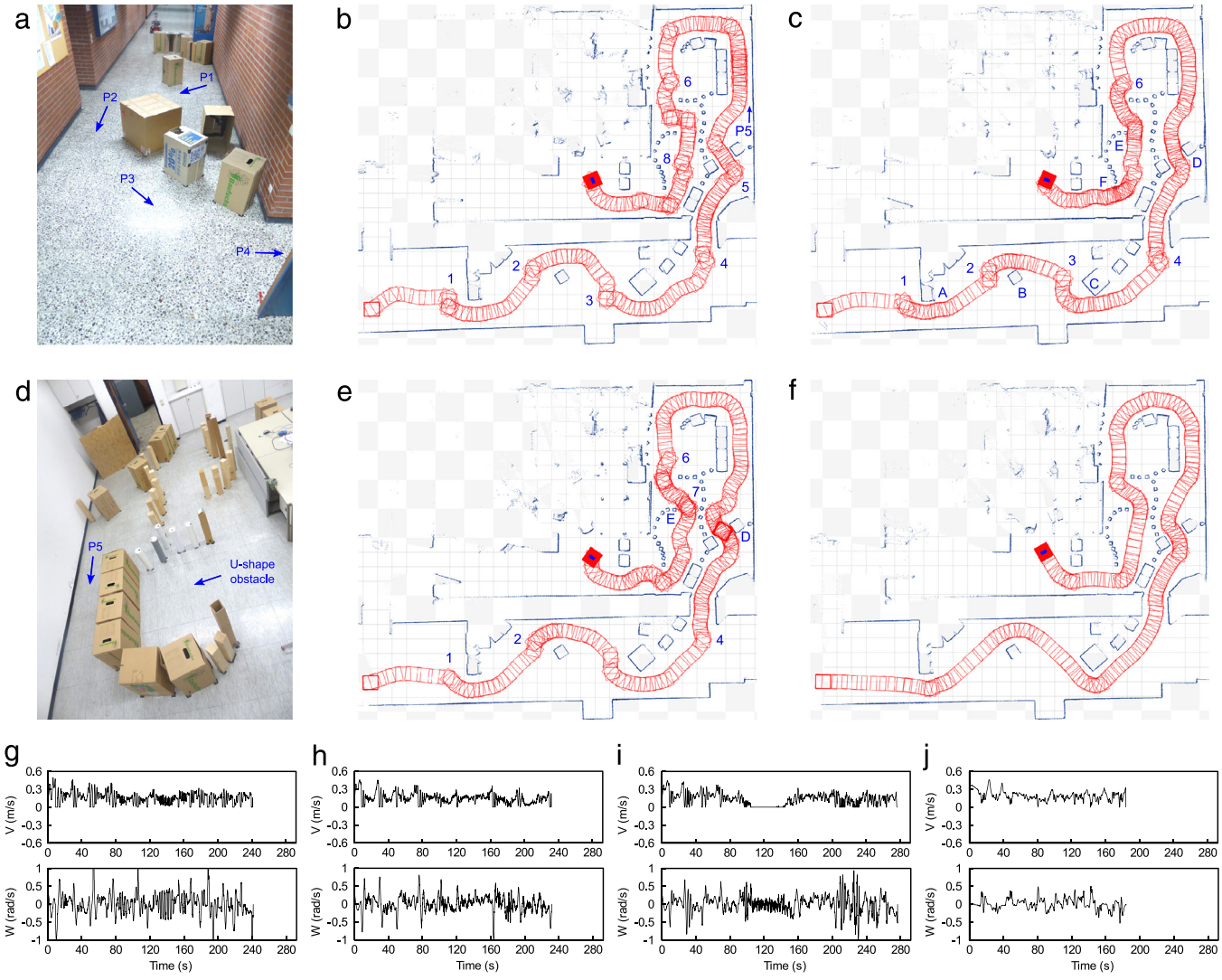

Fig. 10. Experiment 4. (a, d) Environment setup. (b, c, e, and f) Trajectories followed by (b) ND+, (c) SND, (e) CG and (f) TGF. (g–j) Velocity profile for (g) ND+, (h) SND, (i) CG and (j) TGF.

We would like to note that the performance of the methods in this experiment will not be compared in Section 4.4.

4.3. Performance measures

In order to evaluate the effectiveness of the proposed TGF method, and to compare the performance of the TGF to that of the ND variants, we use the following metrics:

(1) **Time to reach the goal** (*TG*): The total time a robot takes to complete the mission [43]. For a high performance, it is preferable to have a low *TG*.

(2) **Path length** (*PL*): The total distance traveled from the starting to the goal locations. Shorter path is desirable to have a better performance. Let $(x_i, f(x_i))$, $(x_t, f(x_t))$ be the initial and target locations, *PL* is defined as [44]:

$$PL = \int_{x_i}^{x_t} (1 + (f'(x))^2)^{\frac{1}{2}}\, dx. \tag{39}$$

(3) **Curvature change** (*CC*): The curvature change is useful for detecting oscillations in the trajectory. It is defined as [43]:

$$CC = \int_0^{TG} \left|k'(t)\right|\, dt, \quad k(t) = \left|\frac{w(t)}{v(t)}\right| \tag{40}$$

where $k(t)$ represents the curvature. Having a low *CC* is desirable for achieving a less oscillatory motion. For comparison purposes, we divide *CC* by the total execution time *TG* in our calculations (average CC).

(4) **Zero crossings of the rotational speed curve** (*ZC*): The total number of zero crossings along the angular speeds curve. It measures how many times the robot changes the direction of motion. Having less *ZC* indicates less changes in the steering angle, and thus more stable motion control.

(5) **Linear and rotational jerk costs** (*LJ* and *AJ*): Jerk is associated with sudden changes in the actuator forces [45]. Hence, it provides a measure of smoothness in the robot's speed as well as steering [46]. Given v and w, we define the linear and rotational jerk costs as follows:

$$LJ = \frac{1}{TG}\int_0^{TG} [\ddot{v}(t)]^2\, dt \tag{41}$$

$$AJ = \frac{1}{TG}\int_0^{TG} [\ddot{w}(t)]^2\, dt. \tag{42}$$

For a smoother behavior, trajectories that minimize *LJ* and *AJ* are desirable.

(6) **Lateral stress** (*LS*): It is directly related to the vehicle dynamics since it measures the centrifugal force acting on the

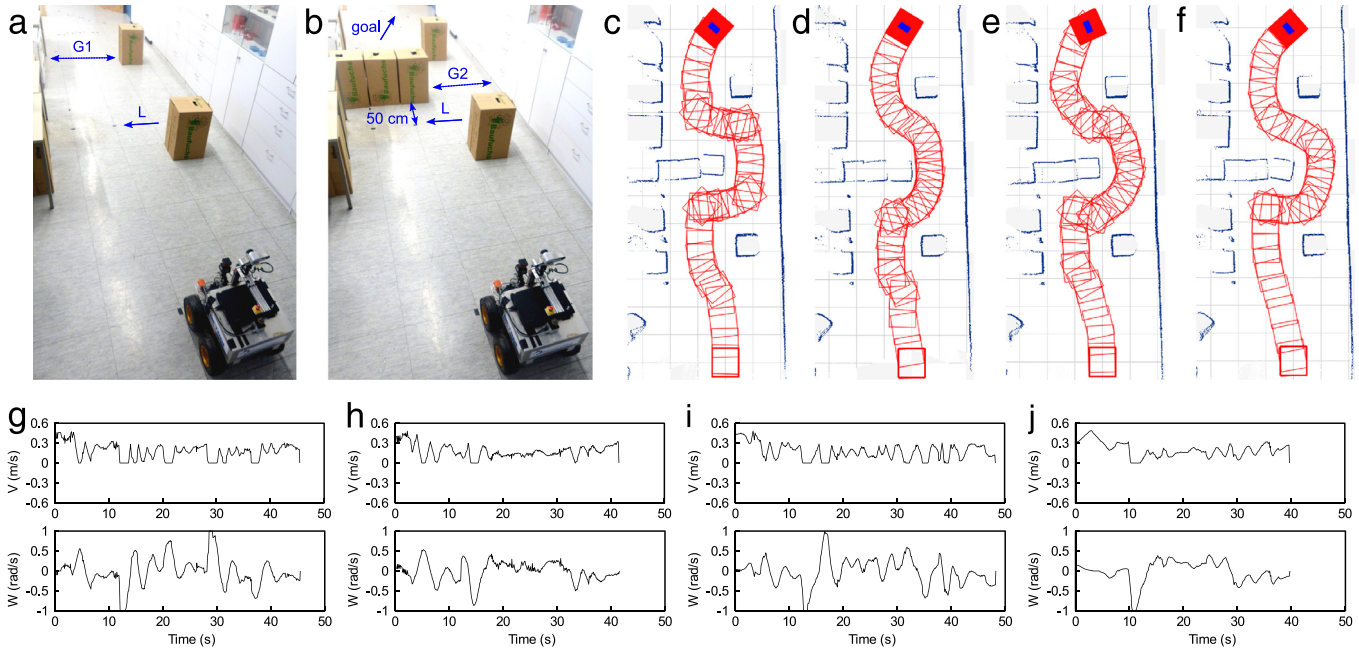

Fig. 11. Experiment 5. (a, b) Environment setup. (c–f) Trajectories followed by (c) ND+, (d) SND, (e) CG and (f) TGF. (g–j) Velocity profile for (g) ND+, (h) SND, (i) CG and (j) TGF.

robot. The straighter the trajectory, the lower *LS* will be. The lateral stress is defined by [43]:

$$LS = \int_0^{TG} \frac{v(t)^2}{r(t)}\, dt, \quad r(t) = 1/k(t). \tag{43}$$

(7) **Tangential stress** (*TS*): It is, like *LS*, directly related to the vehicle dynamics and useful to detect rapidly changing accelerations and decelerations:

$$TS = \int_0^{TG} |\dot{v}(t)|\, dt. \tag{44}$$

For having a more stable vehicle motion, low *TS* is required.

(8) **Risk with respect to obstacles** (*RO*): The risk measure reflects the proximity to obstacles through the entire mission. Let $d_{\min}(t)$ be the distance to the obstacle point closest to the robot at time t, *RO* is then defined by [43]:

$$RO = \int_0^{TG} \frac{1}{d_{\min}(t)}\, dt. \tag{45}$$

(9) **Number of Collisions** (*NC*): Number of collisions per mission [47].

Notice that a division by zero can be encountered while computing *CC*, *LS*, or *RO*. To avoid this problem, we add ϵ to the denominator in Eqs. (40), (43), and (45), where ϵ is any small value (we set $\epsilon = 0.001$ in our calculations).

4.4. Evaluation and discussion

Based on the metrics described in Section 4.3, a performance evaluation of the proposed TGF method as well as a comparison to the ND variants was performed. Table 2 shows the results obtained, where it can be seen that our proposed TGF method outperforms the ND variants in all measures examined. Having a closer look at the *CC*, *ZC*, *LJ*, and *AJ* measures, we can observe significant differences in performance. However, in experiment 1, *LS* and *RO* measures corresponding to ND+ were slightly lower than those corresponding to the TGF. This is due to the fact that under the low safety condition (there are obstacles within D_s on both sides of the closest gap), ND+ tends to move across the center of the gap regardless of its width. This resulted in following a longer path made up almost of straight segments (Fig. 7(b)), that made *LS* and *RO* values lower than TGF, but with a much higher *CC*, *LJ*, and *AJ* costs. Moreover, the trajectory created by SND has lower *PL* and *LS* values compared to the path generated by TGF. Indeed, this is due to the fact that the robot moved very close to obstacles (Fig. 7(c)) which resulted in a shorter and less curved path, but with a much higher *RO* value.

A common drawback of the ND variants can be observed in narrow passages whenever a sudden change in the width of the passage occurs (traveling from relatively wide to narrow gaps) or at sharp corners. For example, look at the point labeled 1 in Fig. 9(b)–(d) and the points marked as 1–8 in Fig. 10(b), (c), and (e). This sudden change causes the robot to highly reduce the speed (mostly zero) and sharply change the direction of motion. The speed acquired before turning increases the problem and may lead to wheel skidding. The values of *CC*, *ZC*, *AJ*, and *TS* reflect this behavior. Basically, this limitation is due to the use of the APF concept to avoid collision with nearby obstacles while progressing towards the goal/closest gap. By this means, getting closer to one side of a narrow passage experiences a strong avoidance angle pushing the robot away by doing a sharp turn, and the process repeats with the other side. Usually, these turn changes excite the robot into oscillatory and unstable motion, particularly if the robot is moving at higher speeds. Moreover, this behavior increases the execution time since turn maneuvers can only be performed at reduced speed.

It is worth mentioning the differences in execution between the ND variants. Among the three tested methods, ND+ seems to be safer and manages harder situations. It was able to successfully drive the robot and reach the goal in all experiments with fewer collisions and lower *RO* value. However, SND and CG are smoother as they have lower *LJ* and *AJ* values (see the behavior when passing passage P5 in experiment 4). Generally speaking, SND and CG are more sensitive to the structure of the environment. For example, when negotiating a narrow passage with a large difference in the number of threats on its sides, SND usually fails (e.g. experiments 2 and 3), collides with obstacles (e.g. D–F in Fig. 10(c)), or gets close to the side having fewer threats (e.g. A and C in Fig. 7(c), A–C in Fig. 10(c)). Using a different proximity weight in the CG method

Table 2
Performance evaluation of the TGF navigation method for experiments 1–4.

Exp.	Method	*TG*	*PL*	*CC*	*ZC*	*LJ*	*AJ*	*LS*	*TS*	*RO*	*NC*
1	ND+	93	10.27	128.25	40	1.42	18.01	**0.84**	8.16	**515.87**	0
	SND	97	**8.75**	13.53	46	0.59	4.21	0.99	6.73	1002.15	0
	CG	117	10.56	121.83	50	0.61	2.84	1.20	11.02	832.02	0
	TGF	**85**	8.96	**1.60**	**10**	**0.13**	**0.46**	1.05	**3.49**	579.70	0
2	ND+	75	8.19	186.37	52	1.95	15.49	1.11	9.96	669.48	1
	SND	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
	CG	75	8.04	213.26	54	1.43	9.29	1.00	10.14	924.68	1
	TGF	**52**	**7.17**	**2.17**	**8**	**0.18**	**0.86**	**0.75**	**2.53**	**445.84**	0
3	ND+	184	16.39	160.01	126	1.55	105.54	2.06	18.42	2635.02	1
	SND	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
	CG	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail
	TGF	**131**	**15.19**	**12.35**	**20**	**0.23**	**1.66**	**1.78**	**6.83**	**1478.25**	0
4	ND+	241	36.80	131.90	121	1.50	8.71	4.91	34.41	1193.48	0
	SND	232	34.10	82.56	143	0.85	6.41	3.97	24.69	2980.42	3
	CG	277	36.04	143.74	215	0.84	9.61	4.49	27.54	41 404.00	1
	TGF	**184**	**33.72**	**9.41**	**38**	**0.26**	**1.07**	**2.72**	**11.09**	**1008.60**	0

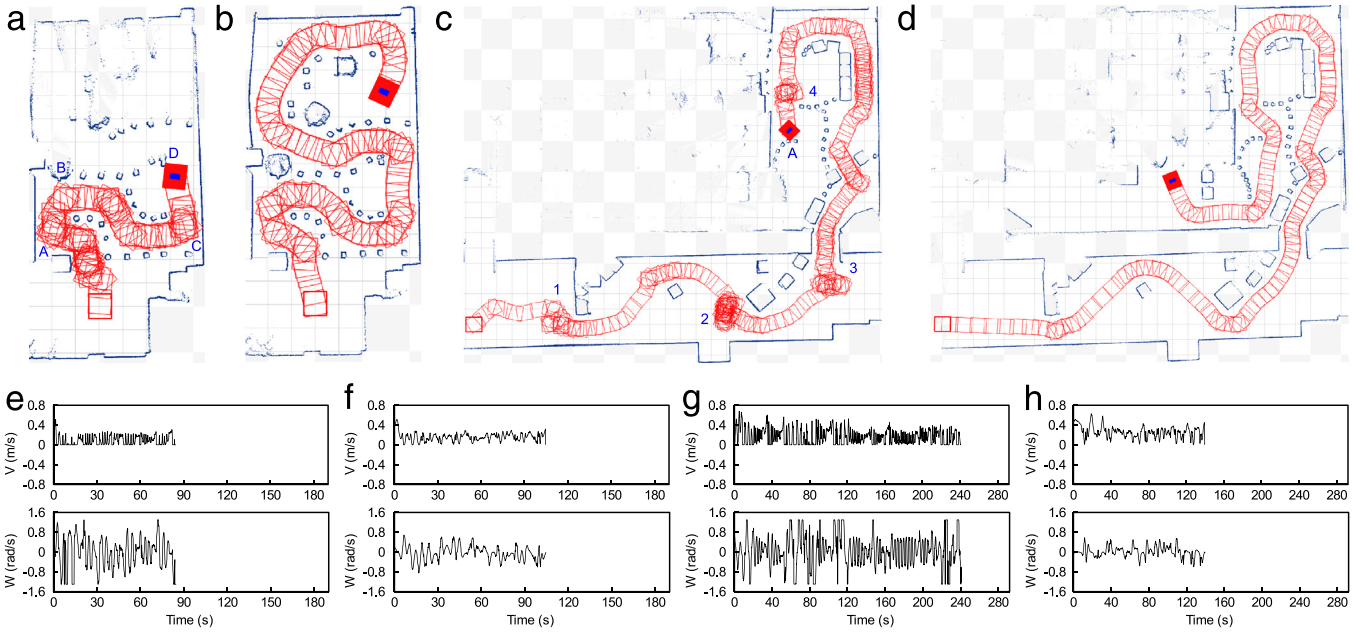

Fig. 12. Experiments 3 and 4 using (0.7 m/s, 1.3 rad/s) maximum speeds. (a, b) Trajectories followed by (a) ND+ and (b) TGF for experiment 3. (c, d) Trajectories followed by (c) ND+ and (d) TGF for experiment 4. (e, f) Velocity profile for (e) ND+ and (f) TGF in experiment 3. (g, h) Velocity profile for (g) ND+ and (h) TGF in experiment 4.

enabled the robot to successfully traverse the course in experiment 2, but with more oscillatory behavior that is reflected by the value of the *CC*. Moreover, it is still prone to collision as occurred in experiment 3.

The increased smoothness and stability of the TGF method enables the robot to safely travel in very dense environments at higher speeds. In order to demonstrate this capability, we carried out experiments 3 and 4 again after setting the maximum linear speed to 0.7 m/s and the maximum rotational speed to 1.3 rad/s. Both experiments were conducted using TGF and ND+ methods (ND+ has been chosen, among ND variants, since it showed the ability to manage harder situations). By applying ND+ in experiment 3, the robot did not complete the course after hitting obstacles A, B, and C, and finally pushing over the obstacles marked as D after 85 s (see Fig. 12(a) and (e)). Using the TGF method, the robot successfully navigated the course and reached the goal only in 105 s (Fig. 12(b) and (f)). In experiment 4, the robot pushed over the obstacles marked as A and came to a full stop after 241 s running the ND+ algorithm (Fig. 12(c) and (g)), whereas TGF managed to successfully drive the robot towards the goal in 140 s (Fig. 12(d) and (h)). Furthermore, when we applied ND+, the motion was oscillatory and unstable. For example, see the trajectory at the points labeled 1–4 in Fig. 12(c).

One can also notice that the TGF method results in a more human-like motion behavior than the ND techniques. This can be interpreted from their reaction to nearby obstacles while progressing towards the closest gap. Using the TGF method, each obstacle generates an avoidance angle which points directly towards the closest gap or parallel to the obstacle in the direction closer to the gap. Hence, all these obstacles contribute in heading the robot towards the closest gap. By using the ND variants, on the other hand, each obstacle point falling within D_s (ND+ only considers the closest two, whereas SND and CG consider all) causes an avoidance angle which points directly away from the obstacle regardless of the direction towards the closest gap. This avoidance angle is weighted by the relative proximity of the obstacles in the SND and CG methods (each one uses different weights). The direction of motion is then adjusted by the total averaged (ND+) or weighted (SND, CG) avoidance angle.

As we have pointed out in Section 4.2, for the sake of a fair comparison, all experiments were carried out using the ND motion commands. In order to show the effect of the new proposed

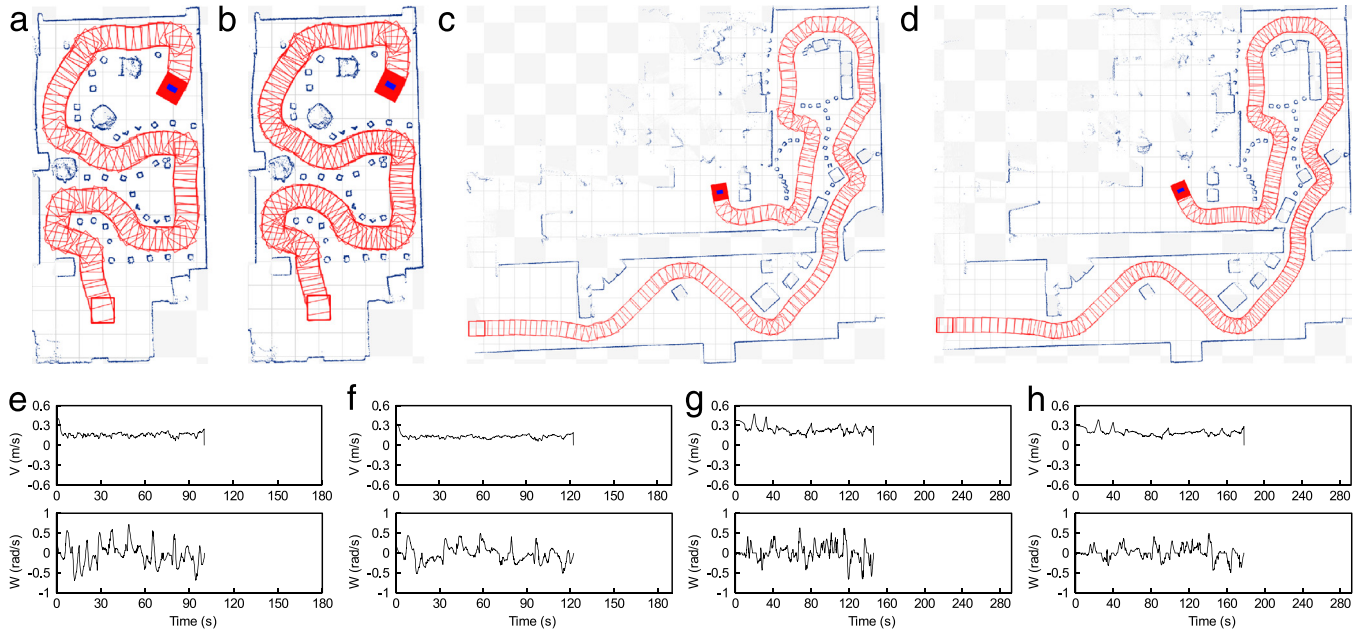

Fig. 13. Experiments 3 and 4 using the TGF motion commands. (a, c) Trajectories followed by the TGF method using (0.5 m/s, 1 rad/s) velocity limits in (a) experiment 3 and (c) experiment 4. (b, d) Trajectories followed by the TGF method using (0.4 m/s, 0.8 rad/s) velocity limits in (b) experiment 3 and (d) experiment 4. (e, g) Velocity profiles for the TGF method using (0.5 m/s, 1 rad/s) velocity limits in (e) experiment 3 and (g) experiment 4. (f, h) Trajectories followed by the TGF method using (0.4 m/s, 0.8 rad/s) velocity limits in (f) experiment 3 and (h) experiment 4.

Table 3
Performance evaluation of the TGF motion commands for experiments 3 and 4. As a reference, the results of the ND motion commands from Table 2 are listed, too.

Exp.	Speeds	*TG*	*PL*	*CC*	*ZC*	*LJ*	*AJ*	*LS*	*TS*	*RO*	*NC*
3 (ND motion)	(0.5, 1.0)	131	15.19	12.35	20	0.23	1.66	1.78	6.83	1478.25	0
3 (TGF motion)	(0.5, 1.0)	100	15.65	1.67	21	0.06	2.47	3.27	3.62	1119.05	0
3 (TGF motion)	(0.4, 0.8)	122	15.46	1.18	7	0.03	1.18	2.42	2.77	1340.76	0
4 (ND motion)	(0.5, 1.0)	184	33.72	9.41	38	0.26	1.07	2.72	11.09	1008.60	0
4 (TGF motion)	(0.5, 1.0)	146	34.05	0.91	30	0.07	2.30	4.59	4.7	803.57	0
4 (TGF motion)	(0.4, 0.8)	178	33.73	0.62	20	0.02	0.88	3.45	3.42	885.50	0

motion commands on the performance of the TGF, we carried out experiments 3 and 4 again but using the TGF motion commands. At first, we used the same maximum velocity limits that we used in the original setup (0.5 m/s, 1 rad/s). Then, we limited these velocities to (0.4 m/s, 0.8 rad/s). Surprisingly, the execution time was smaller than that of the ND motion commands using both velocity limits. Fig. 13 shows the trajectories and the velocity profiles, and Table 3 shows the performance evaluation. We observe a clear performance improvement of the TGF over the ND motion commands in terms of *TG*, *CC*, *LJ*, and *TS*. We believe that this improved performance is due to the fact that the TGF motion commands are directly derived from the kinematic model of the mobile robot with proven control stability. According to the other performance metrics, no significant differences between both motion commands can be observed.

5. Conclusions and future work

We have presented the Tangential Gap Flow (TGF) Navigation method for dense and complex environments. The TGF method drives the robot faster with less-oscillatory motion than the well-known approaches designed to operate in such environments (e.g. the Nearness-Diagram (ND) Navigation methods). This is achieved by introducing two concepts: *tangential* and *gap flow* navigation. The heart of both concepts is to exploit the information obtained from analyzing the structure of obstacles (gap analysis) while generating the avoidance maneuver. The *tangential navigation* drives the robot parallel to the tangent of the obstacle boundary, whereas the *gap flow navigation* points the robot towards the free area between obstacles on both sides of the robot's heading. Both concepts perform the avoidance while emphasizing the progress towards the goal/closest gap. The smoothness and the stability of the trajectories generated by the TGF have been improved by taking all nearby obstacle points into consideration while determining the avoidance maneuver. Moreover, the stability of the control system adopted was proven using the Lyapunov theorem. A performance evaluation was also performed to quantitatively estimate the power of the TGF method over the ND variants.

This paper focuses on improving the trajectories generated by the ND navigation techniques. Other advantages that are common between our approach and the ND methods, such as solving trapping situations due to U-shaped obstacles, are not addressed in this paper. A detailed description of these advantages including a comparison with other existing techniques is presented in [2].

We have seen that the TGF solves the motion problem using only the current sensor data as explained in Section 3. Therefore, trap situations or cyclic motions may arise. To solve such a problem, the TGF should be used in conjunction with a planner such as that proposed in [48] and [49]. In this regard, these undesirable situations can be avoided while still using a high-frequency navigation module which is necessary in unstructured and dynamic environments.

After testing the TGF method, we are satisfied with the results obtained to date. There is, however, room for improvement given the complexity of finding a solution suitable for all scenarios and

all types of environments. As we have pointed out in Section 3.2, determining Ψ_{vg} is mainly based on detecting obstacles within D_s. Although the performance of the TGF is stable against changes of the value of D_s, we see that it can be improved by adaptively choosing D_s based on the structure of the environment, robot shape, and motion constraints. Furthermore, the TGF controller proposed in Section 3.3 is designed assuming a flat ground, which is not the case in disaster scenarios. It is of interest to adapt our controller by taking into account navigation over uneven terrains.

Acknowledgment

This work was supported by the German Academic Exchange Service (DAAD), grant no. 57076385.

Appendix. Index to multimedia extensions

The multimedia extensions to this article are available at:

(1) http://getwww.uni-paderborn.de/research/videos/tgf1
(2) http://getwww.uni-paderborn.de/research/videos/tgf2
(3) http://getwww.uni-paderborn.de/research/videos/tgf3.

Ext.	Media	Description
1	Video	The accompanying video material shows scenes of all experiments presented in Section 4: (1) Experiments 1–5 using the *ND motion commands*. (2) Experiments 3 and 4 using higher velocity limits (0.7 m/s, 1.3 rad/s). (3) Experiments 3 and 4 using the *TGF motion commands* with two different velocity setups (0.5 m/s, 1.0 rad/s), (0.4 m/s, 0.8 rad/s).

References

[1] S. Hirose, E.F. Fukushima, Snakes and strings: New robotic components for rescue operations, Int. J. Robot. Res. 23 (4–5) (2004) 341–349.
[2] J. Minguez, L. Montano, Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios, IEEE Trans. Robot. Autom. 20 (1) (2004) 45–59.
[3] J. Minguez, The obstacle-restriction method for robot obstacle avoidance in difficult environments, in: IROS, 2005, pp. 2284–2290.
[4] J.W. Durham, F. Bullo, Smooth nearness-diagram navigation, in: IROS, France, 2008, pp. 690–695.
[5] 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.
[6] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robot. Res. 5 (1) (1986) 90–98.
[7] Y. Koren, J. Borenstein, Potential field methods and their inherent limitations for mobile robot navigation, in: ICRA, CA, 1991, pp. 1398–1404.
[8] 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.
[9] M. Mujahed, D. Fischer, B. Mertsching, Safe gap based (SG) reactive navigation for mobile robots, in: ECMR, Barcelona, Spain, 2013, pp. 325–330.
[10] J.-C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1991.
[11] S.M. LaValle, Planning Algorithms, Cambridge University Press, 2006.
[12] J. Borenstein, Y. Koren, Real-time obstacle avoidance for fast mobile robots, IEEE Trans. Syst. Man Cybern. 19 (5) (1989) 1179–1187.
[13] L. Montano, J.R. Asensio, Real-time robot navigation in unstructured environments using a 3D laser rangefinder, in: IROS, France, Vol. 2, 1997, pp. 526–532.
[14] L. Chengqing, M.H. Ang, H. Krishnan, L.S. Yong, Virtual obstacle concept for local-minimum-recovery in potential-field based navigation, in: ICRA, San Francisco, USA, 2000, pp. 983–988.
[15] J. Borenstein, Y. Koren, The vector field histogram - fast obstacle avoidance for mobile robots, IEEE Trans. Robot. Autom. 7 (3) (1991) 278–288.
[16] A. Manz, R. Liscano, D. Green, A comparison of real-time obstacle avoidance methods for mobile robots, in: Experimental Robotics II, Toulouse, France, 1991, pp. 299–316.
[17] J. Ren, K. McIsaac, R. Patel, Modified Newtons method applied to potential field based navigation for nonholonomic robots in dynamic environments, Robotica 26 (2008) 117–127.
[18] D. Panagou, Motion planning and collision avoidance using navigation vector fields, in: ICRA, Hong Kong, China, 2014, pp. 2513–2518.
[19] J. Barraquand, J.C. Latombe, Robot motion planning: A distributed representation approach, Int. J. Robot. Res. 10 (6) (1991) 628–649.
[20] E. Rimon, D.E. Koditschek, Exact robot navigation using artificial potential functions, IEEE Trans. Robot. Autom. 8 (5) (1992) 501–518.
[21] 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: ICRA, Seattle, USA, 2015, pp. 2347–2354.
[22] H. Secchi, R. Carelli, V. Mut, Discrete stable control of mobile robots with obstacles avoidance, in: ICAR, Budapest, Hungary, 2001, pp. 405–411.
[23] A. Ferreira, F.G. Pereira, R.F. Vassallo, M. Filho, T.F. Filho, An approach to avoid obstacles in mobile robot navigation: The tangential escape, SBA Soc. Bras. Automat. 19 (4) (2008) 395–405.
[24] R. Simmons, The curvature-velocity method for local obstacle avoidance, in: ICRA, USA, 1996, pp. 3375–3382.
[25] 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.
[26] D. Fox, W. Burgard, S. Thrun, The dynamic window approach to collision avoidance, IEEE Robot. Autom. Mag. 4 (1) (1997) 23–33.
[27] 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.
[28] P. Fiorini, Z. Shiller, Motion planning in dynamic environments using velocity obstacles, Int. J. Robot. Res. 17 (7) (1998) 760–772.
[29] F. Large, C. Laugier, Z. Shiller, Navigation among moving obstacles using the NLVO: Principles and applications to intelligent vehicles, Auton. Robots 19 (2) (2005) 159–171.
[30] A. Wu, J.P. How, Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles, Auton. Robots 32 (3) (2012) 227–242.
[31] T. Fraichard, H. Asama, Inevitable collision states - a step towards safer robots? Adv. Robot. 18 (10) (2004) 1001–1024.
[32] A. Lawitzky, A. Nicklas, D. Wollherr, M. Buss, Determining states of inevitable collision using reachability analysis, in: IROS, USA, 2014, pp. 4142–4147.
[33] J. Berg, M.C. Lin, D. Manocha, Reciprocal velocity obstacles for real-time multi-agent navigation, in: ICRA, Pasadena, CA, 2008, pp. 1928–1935.
[34] D. Bareiss, J. Berg, Generalized reciprocal collision avoidance, Int. J. Robot. Res. 34 (12) (2015) 1501–1514.
[35] J. Jin, Y. Kim, S. Wee, N. Gans, Decentralized cooperative mean approach to collision avoidance for nonholonomic mobile robots, in: ICRA, USA, 2015, pp. 35–41.
[36] Z. Shiller, S. Sharma, High speed on-line motion planning in cluttered environments, in: IROS, Vilamoura, Portugal, 2012, pp. 596–601.
[37] 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.
[38] 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.
[39] I. Ulrich, J. Borenstein, Vfh+: Reliable obstacle avoidance for fast mobile robots, in: ICRA, 1998, pp. 1572–1577.
[40] M. Aicardi, G. Casalino, A. Bicchi, A. Balestrino, Closed loop steering of unicycle-like vehicles via Lyapunov techniques, IEEE Robot. Autom. Mag. 2 (1995) 27–35.
[41] F.G. Pereira, M.C.P. Santos, R.F. Vassallo, A nonlinear controller for people guidance based on omnidirectional vision., in: IROS, San Francisco, USA, 2011, pp. 3620–3625.
[42] B. Gerkey, contributors, Karto mapping library, April 2010. http://wiki.ros.org/karto/ (accessed: 03.08.15).
[43] 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.
[44] N. Munoz, J. Valencia, N. Londono, Evaluation of navigation of an autonomous mobile robot, in: PerMIS, 2007, pp. 15–21.
[45] P. Freeman, Minimum Jerk Trajectory Planning for Trajectory Constrained Redundant Robots (Ph.D. thesis), Washington University in St. Louis, 2012.
[46] J. Rosenblatt, Damn: A Distributed Architecture for Mobile Navigation (Ph.D. thesis), Robotics Institute, Carnegie Mellon University, 1997.
[47] J. Minguez, Robot Obstacle Avoidance Papers Using Experiments, Tech. Rep., Robotics European Robotics Research Network (EURON), 2008.
[48] J. Minguez, L. Montano, T. Simeon, R. Alami, Global nearness diagram navigation (GND), in: ICRA, Seoul, Korea, 2001, pp. 33–39.
[49] C. Stachniss, W. Burgard, An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments, in: IROS, 2002, pp. 508–513.

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 full 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.