The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Closest Gap Based (CG) Reactive Obstacle Avoidance Navigation for Highly Cluttered Environments

Muhannad Mujahad, Dirk Fischer, Bärbel Mertsching, and Hussein Jaddu

Abstract*— A new reactive collision avoidance approach for mobile robots moving in cluttered and complex environments was developed and implemented. The novelty of this approach lies in the creation of a new method for analyzing openings in front of the robot that highly reduces their number when compared with the Nearness-Diagram Navigation (ND) technique, particularly in complex scenarios. Moreover, the angular width of the chosen (selected) gap with respect to the robot vision is taken into consideration. Consequently, oscillations are alleviated, the computational complexity is reduced and a smoother behavior will be achieved. Our technique adjusts the motion law proposed in the Smooth Nearness-Diagram Navigation (SND) method to generate safer paths for the robot by considering the ratio of threats on its sides and applying stricter deviation against an obstacle as it gets closer to the robot. Hence, the problem of deadlock occurring in narrow corridors, with high threats on one side and low threats on the other, is solved without affecting the smoothness behavior. Simulation and experimental results demonstrate the power of the proposed approach.

I. Introduction

A lot of interest has been given to autonomous mobile robots in the past few years. The aim of these robots is to build physical systems that can navigate safely through an environment without human intervention [1]. This is due to the fact that many applications in real life are difficult to be carried out by humans such as: industrial applications [2], rescue [3], military [25, 26] and exploration [4], among others. Usually, the environment the mobile robot moves through in order to carry out the tasks specified for an application is unknown. In this case, the navigation challenge is to find a method to guide a robot to a given goal while avoiding dynamic obstacles which can be distributed randomly in its way such as: humans, tables, stones and even other robots operating in the same area. Motion planning algorithms [27] rely on accurate, static models of the environments. Thus, they often fail if the environment is unknown to the robot or unpredictable obstacles block its path. To overcome this limitation the motion techniques must depend on sensors detecting instantaneous changes in the environment or obstacles appearing periodically. This can be achieved by reactive navigation methods.

Many existing reactive navigation methods have problems in dealing with dense and cluttered environments, which is usually the case in most robotic applications. Some drawbacks of these approaches are: the local minima problem, deadlock, reaching an oscillation in behavior and the computational complexity. This paper overcomes all these shortcomings through introducing a new local reactive navigation scheme depending on two previous works; the Nearness-Diagram (ND) and the Smooth Nearness-Diagram (SND) navigation methods [16, 19].

We call the *Closest Gap* (CG) the basis for analyzing *gaps* in our design (a gap is a potential free path wide enough for the robot to move through). CG is able to find gaps that are directly in front of the robot and cancels others that are not necessary. Hence, the computational complexity is reduced, oscillations are alleviated and a smooth behavior will be achieved. The navigable gap closest to the goal is then chosen from among these gaps taking into consideration the angular width of the robot vision. The main contributions of our approach in calculating the motion commands refer to considering the ratio of threat counts on the two sides of the robot and providing stricter behavior against the closest obstacles. As a consequence, the robot is capable of avoiding the SND deadlock problem, which occurs in narrow corridors where the difference in the number of threats on its sides is high, while keeping the smoothness behavior. Hence, a robust reactive navigation algorithm for highly cluttered environments is obtained.

This paper discusses related work in Section II, and presents the reactive obstacle avoidance method in Section III. In Sections IV and V, we show the simulation and experimental results. Finally, Section VI highlights our conclusions.

II. Related Work

In this section, we will survey only the reactive navigation algorithms. According to the global techniques, the reader is directed to [27] for an extended knowledge and taxonomy of these methods.

Early work in this topic includes the Artificial Potential Fields [5], which assumes that obstacles exert a repulsive force while the target asserts an attractive force on the robot. The resultant vector sums of these forces are used to compute the robot's steering direction. The Potential Field approach has been widely used by a large number of researchers (e.g. [6 - 10]). Although this approach is considered fast and computationally efficient, it suffers from many shortcomings such as: production of local minima trapping the robot and failure to pass between two close obstacles [12]. The Vector Field Histogram (VFH) [13] is then introduced drawing a two-dimensional grid based on

Manuscript submitted March 10, 2010.

M. Mujahed and H. Jaddu are with the Faculty of Engineering, Al-Quds University, P.O. Box 20002, Abu Dies, Jerusalem, Palestine, via Israel, muh_mujahed@yahoo.com, jaddu@eng.alquds.edu.

D. Fischer and B. Mertsching are with the Dept. of Electrical Engineering, GET-Lab, University of Paderborn, Pohlweg 47-49, D-33098 Paderborn, Germany, {fischer, mertsching}@get.uni-paderborn.de.

the Certainty Grid concept [11] for obstacle representation and then reducing it to a one-dimensional polar histogram. The direction of the robot is computed by choosing the sector with the least concentration of obstacles. This approach was less likely to be trapped in local minima. However, the VFH and even the enhancements proposed in [14, 15] present the difficulty to move between close obstacles due to the tuning of a threshold which depends on the obstacle density.

The Nearness-Diagram Navigation (ND) method [16] overcomes all previously mentioned limitations on potential field and polar histogram approaches through densely analyzing the information from laser sensors and then determining the motion commands. By this method, the robot can navigate safely through cluttered and complex environments. The ND divides navigation behavior into five situations to take action as required based on the situated-activity paradigm design [17]. Afterwards, authors of ND reformulated the motion laws and added a sixth situation to lead to the ND+ method [18]. Then, [Durham and Bullo] further developed the ND+ to produce a smoother navigation technique: The Smooth Nearness-Diagram (SND) Navigation [19], which generates a single motion law by considering all nearby obstacles.

All previously mentioned techniques can be classified as stated in [20] as directional approaches where the navigation problem is divided into two parts. First, sensory information is analyzed for finding a proper direction. Second, the robot is controlled to move towards that direction. The CG navigation method belongs to this group of approaches. Other methods take the robot dynamics and kinematics into consideration and depend on velocity space to compute the motion commands. The most popular ones are: The curvature-velocity [21] and the dynamic window (DW) approaches [22]. An enhancement of the DW approach was introduced in [23], which generates a collision-free motion by considering the information of a real map of the environment plus data from the sensors. Then, the DW method was further improved by [24] to accommodate holonomic and non- holonomic robots in order to better deal with local minima problems.

Although velocity space approaches have faster and smoother behavior than directional methods, the local minima problem can appear. Thus, it is better to use directional approaches in complex and cluttered environments.

III. The Reactive Obstacle Avoidance Method

We explain in this section the Closest Gap Navigation method (CG) for avoiding obstacles in complex and cluttered environments. The CG works as follows: first, the information from the laser rangefinder sensor is periodically analyzed to identify the gaps surrounding the robot as explained in Section III-B. In order to reach the goal, the closest navigable gap is chosen. The direction of motion towards this gap is determined as described in Section III-C. Finally, the real time deflection from obstacles while moving towards the goal is introduced in Section III-D.

A. Definitions

In this section we explain some definitions introduced in [19]. The positive x axis is in front of the robot and the positive y axis is the normal on its left side. The angles always have an absolute value less than π. Negative angles are on the right side of the robot whereas positive angles are on the left.

Let $\mathbb{S}$ be the unit circle attached to the robot's reference frame. For any two angles $\alpha, \beta \in \mathbb{S}$, the angular distance between them relative to the robot is $\text{dist}(\alpha,\beta) = \min\{\text{dist}_{cc}(\alpha,\beta), \text{dist}_{c}(\alpha,\beta)\}$ where $\text{dist}_{cc}(\alpha,\beta) = (\beta-\alpha) \mod 2\pi$ and $\text{dist}_{c}(\alpha,\beta) = (\alpha-\beta) \mod 2\pi$. Sometimes, an angle θ may become greater than π or less than $-\pi$ during calculations. In order to map this angle into the right value in $[-\pi,\pi[$, the projection function is defined as:

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

Finally, the saturation function is used to limit a value between two boundaries. Assume that a < b, the *sat* function is defined as follows:

$$\text{sat}_{[a,b]}(\mathrm{x}) = \begin{cases} a, & \text{if } x \le a, \\ x, & \text{if } a < x < b, \\ b, & \text{if } x \ge b. \end{cases} \tag{2}$$

B. Analyzing Gaps

The main part in analyzing the data perceived by sensors is to identify the *gaps* surrounding the robot. Before explaining the details, assume the following.

1) The first scan point is (0) and the final one is $(n-1)$.
2) (L) is the list of obstacle points perceived (detected).
3) The maximum range of the sensor is denoted by $(d_{\max})$.
4) $\mathrm{d}(A,B)$ returns the distance between points A and B.
5) We will take all scan points into consideration (not divide into sectors) in this algorithm.

The inputs of the algorithm are:

1) The robot location $(\mathrm{x}_{\text{robot}})$ and robot radius R.
2) The maximum range of the sensor $(d_{\max})$.
3) A list (L) of obstacle points where an obstacle is $(\mathrm{O}_{\mathrm{i}}^{\mathrm{L}})$.

The output of the algorithm is the list of gaps detected.

Extracting gaps can be summarized in two steps; the first one implies finding the list of all gaps that are seen by the robot dependent on fetching discontinuities and the other removes unusual gaps from this list. Before going into details of the two steps, let us define two types of discontinuities that occur between two adjacent scans i and j (assuming the original order of scans in the forward and backward loops explained below).

Type 1 discontinuity: Occurs when the difference between the depth measurements of scans i and j exceeds the robot diameter.

$$(d(\mathrm{x}_{\text{robot}}, \mathrm{O}_{\mathrm{j}}^{\mathrm{L}}) - d(\mathrm{x}_{\text{robot}}, \mathrm{O}_{\mathrm{i}}^{\mathrm{L}})) > 2R.$$

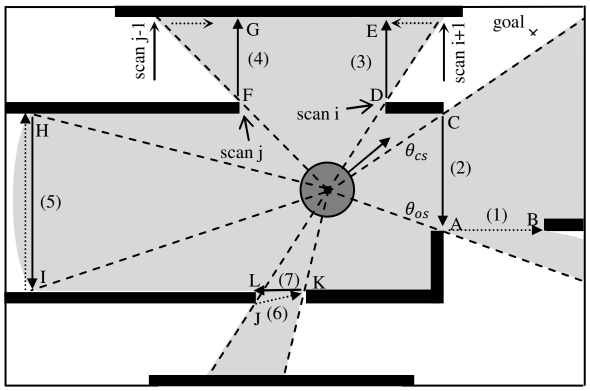

Fig. 1. Analyzing gaps by the CG method. At first, gaps that are labeled from 1 to 7 are detected. Then, gap 1 and gap 6 are eliminated.

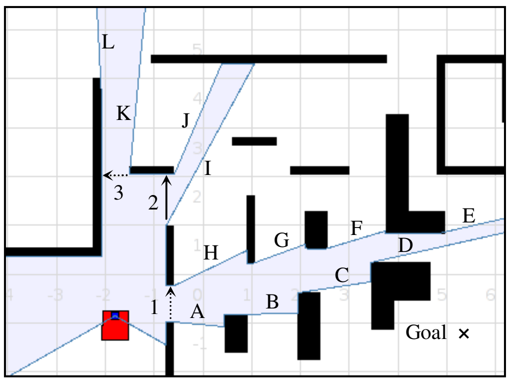

Fig . 2. Analyzing gaps showing advantages of the CG method over the SND approach.

Type 2 discontinuity: Occurs if one of the two measurements returns the maximum sensor range.

$$\mathrm{d}(\mathrm{x}_{\mathrm{robot}}, \mathrm{O}_{\mathrm{j}}^{\mathrm{L}}) = d_{\max} \text{ AND } \mathrm{d}(\mathrm{x}_{\mathrm{robot}}, \mathrm{O}_{\mathrm{i}}^{\mathrm{L}}) < d_{\max}$$

if $j > i$, a *rising* discontinuity occurs at scan i; else, it will be a *descending* discontinuity at scan j. Type 1 discontinuity has a higher priority than type 2.

Step 1: We scan for gaps twice; first by detecting rising discontinuities while travelling from scan 0 to $n-1$ (forward search) and then through fetching descending discontinuities travelling from scan $n-1$ to 0 (backward search). Assume that the first rising discontinuity occurs at scan number i in the forward search. This scan determines the first side of the gap (e.g. points D and H in Fig. 1). Finding the second side depends on the discontinuity type as shown below.

1) For a type 1 discontinuity:
Let $\mathrm{S}^{+} = \{i+1, \dots, n-1\}$ be the set of all scans after scan i. The second side of the gap will be at scan number $j \in \mathrm{S}^{+}$, which satisfies the shortest distance to the first side; the angular distance travelled must be less than π (e.g. Fig. 1, point E).

$$\mathrm{d}(\mathrm{O}_{\mathrm{i}}^{\mathrm{L}}, \mathrm{O}_{\mathrm{j}}^{\mathrm{L}}) \le \mathrm{d}(\mathrm{O}_{\mathrm{i}}^{\mathrm{L}}, \mathrm{O}_{\mathrm{k}}^{\mathrm{L}}) \text{ AND } \mathrm{dist}_{\mathrm{cc}}(\theta_{\mathrm{i}}, \theta_{\mathrm{j}}) \le \pi$$

where k is any scan number $\in \mathrm{S}^{+}$.

2) For a discontinuity type 2:
Continue scanning after scan i until a descending discontinuity, either type 1 or type 2, is detected at scan number j. In this case, the second side will be at scan $j+1$ (see point I, Fig. 1).

In order to find the rest of forward gaps, resume the process starting from scan $j+1$. This search produces gaps 1, 3, 5 and 6 in Fig. 1.

In the backward search, assume that a descending discontinuity occurs at scan number j. The second side of the gap will be at scan $j+1$ (e.g. points F and I in Fig. 1). In order to find the first side, the following is done:

1) For a detected discontinuity of type 1:
Let $\mathrm{S}^{-} = \{j-1, \dots, 0\}$ be the set of all scans before scan j. The first side of the gap will be at scan number $i \in \mathrm{S}^{-}$, which satisfies the shortest distance to the second side; the angular distance travelled must be less than π (see point G in Fig. 1).

$$\mathrm{d}(\mathrm{O}_{\mathrm{i}}^{\mathrm{L}}, \mathrm{O}_{\mathrm{j+1}}^{\mathrm{L}}) \le \mathrm{d}(\mathrm{O}_{\mathrm{k}}^{\mathrm{L}}, \mathrm{O}_{\mathrm{j+1}}^{\mathrm{L}}) \text{ AND } \mathrm{dist}_{\mathrm{c}}(\theta_{\mathrm{j+1}}, \theta_{\mathrm{i}}) \le \pi$$

where k is any scan number $\in \mathrm{S}^{-}$.

2) For a discontinuity of type 2:
Pass through the scans that come before scan j until a rising discontinuity, either type 1 or type 2, is fetched at scan number i. This scan determines the second side (e.g. Fig. 1, point H). In case of type 2, delete the gap since it is considered in the forward loop.

In order to find the rest of backward gaps, resume the process from scan number $i-1$. Gaps 7, 4 and 2 in Fig. 1 are fetched in this search.

Step 2: After completing the forward and backward loops, we get the list of gaps (G). Eliminate from G every gap that exists inside another gap (e.g. gaps 1 and 6 in Fig. 1), and then remove from the remaining gaps any gap that has a width less than the robot diameter (e.g. gap 7 in Fig. 1). The following steps explain the idea.

1) If $\exists a, b \in \mathrm{G} | (a_{\mathrm{i}} \ge b_{\mathrm{i}} \text{ AND } a_{\mathrm{j}} \le b_{\mathrm{j}})$, then eliminate **a**
2) If $\exists c \in \grave{\mathrm{G}} | \mathrm{d}(c_{\mathrm{i}}, c_{\mathrm{j}}) < 2R$, then eliminate **c**

where x_{i} and x_{j} denote the first and second sides of gap x, $\grave{\mathrm{G}} \in \mathrm{G}$ is the list of remaining gaps after step 1.

To demonstrate the strength of our method for analyzing gaps, we took a snapshot of the Player/Stage simulator which shows how gaps are fetched in the SND and the CG methods (Fig. 2). The SND method detects twelve gaps which are labeled A to L while the CG algorithm returns only one gap labeled number 2. This is done in the CG method as follows: The rising discontinuities A, I and k form the gaps 1, 2 and 3 in the forward search. The gaps which are detected during the backward search from the descending discontinuities L, J and H are deleted in step 2 of the algorithm, since they are contained inside gaps 1, 2 and 3. Gaps 1 and 3 are then deleted since their width is less than the diameter of the robot. It is obvious from the figure that the gaps from A to H do not have to be considered at this point since gap 1 leads to them. Similar situations may arise many times during navigation, particularly in complex environments. Avoiding this decreases the possibility of oscillations that may occur from the great number of unnecessary gaps. Also, this will reduce the computational complexity needed for calculations.

Once the list of gaps is assembled, the *navigable* one closest to the goal is selected. It is identified by selecting the gap with the side closest to the goal (the angle between this side and the goal is the minimum). Then, this gap is checked if it is navigable. If it is not, another gap is selected in the same manner and the process is repeated until a navigable gap is found, or no gaps exist. We refer to the side of the selected gap closest to the goal by the angle θ_{cs} and the other side of this gap by the angle θ_{os}. The closest gap in Fig. 1 is gap 2. The closest side to the goal occurs at point C while the other side is at point A.

Remark 1 (Checking Navigability): To verify whether a gap is navigable: if the goal location lies inside the *closest gap*, we check for an existing path to the goal as stated in the appendix A of the ND paper [16]. If the goal location is not inside the closest gap, we check for an existing path to the middle of the gap that is the point between the first and the second side of the gap.

C. Determining Motion Direction

In this section, we present a procedure to determine the motion direction based on the analysis made in section III-B. First of all, we check if there is a direct and *navigable* way to the goal. If so, we do not look at gaps at all. We only set the motion direction $\theta_{\mathrm{md}} = \theta_{\mathrm{goal}}$. If there is no free way to the goal, the robot shall pass through the *closest gap* assigned in section III-B.

As previously mentioned each gap has two sides, one is to the left of the other. We call it a *left side* of the gap and the other is a *right side*. To go safely through the closest gap as a step towards the goal, we use the two angles defined in [19].

$$\theta_{\mathrm{scs}} = \begin{cases} \theta_{\mathrm{cs}} - \arcsin\left(\frac{R+D_{\mathrm{s}}}{D_{\mathrm{cs}}}\right), & \text{if } \theta_{\mathrm{cs}} \text{ is a left side,} \\ \theta_{\mathrm{cs}} + \arcsin\left(\frac{R+D_{\mathrm{s}}}{D_{\mathrm{cs}}}\right), & \text{if } \theta_{\mathrm{cs}} \text{ is a right side,} \end{cases} \quad (3)$$

$$\theta_{\mathrm{mid}} = \begin{cases} \theta_{\mathrm{cs}} - \mathrm{dist}_{\mathrm{c}}(\theta_{\mathrm{cs}}, \theta_{\mathrm{os}})/2, & \text{if } \theta_{\mathrm{cs}} \text{ is a left side,} \\ \theta_{\mathrm{cs}} + \mathrm{dist}_{\mathrm{cc}}(\theta_{\mathrm{cs}}, \theta_{\mathrm{os}})/2, & \text{if } \theta_{\mathrm{cs}} \text{ is a right side,} \end{cases} \quad (4)$$

where D_{s} and D_{cs} are the safe distance and the distance to the obstacle point creating the side of the *closest gap* closest to the goal from the center of the robot, respectively. We choose θ_{md} based on two things: the location of the goal (if θ_{goal} falls between θ_{cs} and θ_{os}, set $\theta_{\mathrm{md}} = \theta_{\mathrm{goal}}$) and the width of the gap (choose θ_{scs} if the gap is wide or θ_{mid} if it is narrow). The following equation explains that:

$$\theta_{\mathrm{md}} = \begin{cases} \theta_{\mathrm{goal}}, & \text{if } \theta_{\mathrm{rs}} \leq \theta_{\mathrm{goal}} \leq \theta_{\mathrm{ls}}, \\ \theta_{\mathrm{mid}}, & \text{if } \mathrm{dist}(\theta_{\mathrm{cs}}, \theta_{\mathrm{mid}}) < \mathrm{dist}(\theta_{\mathrm{cs}}, \theta_{\mathrm{scs}}), \\ \theta_{\mathrm{scs}}, & \text{otherwise,} \end{cases} \quad (5)$$

where θ_{rs} and θ_{ls} are the angles toward the left and right side of the closest gap.

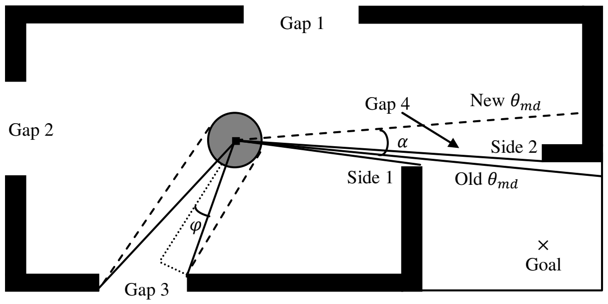

Fig. 3. Modifying θ_{md} according to the angular width of the closest gap with respect to the robot vision.

Our approach adjusts θ_{md} by adding another angle (α) to provide a smoother and safer behavior than SND and ND+. Fig. 3 shows the usefulness of this. It can be seen that four gaps surround the robot. The closest one to the goal is gap 4, which has a narrow width. So, as stated above, the motion direction is through the middle of the gap (θ_{mid}). It is clear that the gap aperture, as seen by the robot, is very small. In this regard, the direction of motion will be nearly towards side (1). The robot will move in this direction till it gets close to the obstacles at this side. Then, the real time obstacle avoidance algorithm introduced hereinafter in section III-D will deflect the direction away from these obstacles. This reduces the smoothness and may cause the robot to collide with obstacles if the robot is fast or the safe distance is short. Gap 3 has an opening angle wide enough to fit the robot diameter. In this case, there is no problem. To solve this drawback, we propose to modify θ_{md} to let the gap fit the robot diameter.

Let us first define the angle φ which is the minimum to fit the radius of the robot:

$$\varphi = \arcsin\left(\frac{R}{D_{\mathrm{ns}}}\right) \quad (6)$$

where R and D_{ns} are the robot radius and the distance from the center of the robot to the gap side closest to the robot.

Now, suppose $\beta = 2\varphi$ is the most narrow angle which fits the robot diameter. The real width of the gap as seen by the robot is defined as follows:

$$w = \begin{cases} \mathrm{dist}_{\mathrm{c}}(\theta_{\mathrm{cs}}, \theta_{\mathrm{os}}), & \text{if } \theta_{\mathrm{cs}} \text{ is a left side,} \\ \mathrm{dist}_{\mathrm{cc}}(\theta_{\mathrm{cs}}, \theta_{\mathrm{os}}), & \text{if } \theta_{\mathrm{cs}} \text{ is a right side,} \end{cases} \quad (7)$$

Now, we define α as:

$$\alpha = \mathrm{sat}_{[0,\beta]}(\beta - w) \quad (8)$$

where the sat operator caps α at 0 when $w \geq \beta$ (since there is no need to modify θ_{md} in this case), at β when w = 0 (the maximum) and $(\beta - w)$ when $0 < w < \beta$.

Finally, we adjust θ_{md} as follows:

$$\theta_{\mathrm{md}} = \begin{cases} \theta_{\mathrm{md}} - \alpha, & \text{if } D_{\mathrm{ls}} < D_{\mathrm{rs}}, \\ \theta_{\mathrm{md}} + \alpha, & \text{otherwise,} \end{cases} \quad (9)$$

where D_{ls} and D_{rs} are the distances to the left and right side of the closest gap, respectively, from the center of the robot.

In Eq. (9) the angle α is added or subtracted dependent on the sides of the closest gap to ensure that θ_{md} will force the robot to move towards the gap.

D. Real Time Reactive Navigation Method

After analyzing sensory information and getting the motion direction, the robot should be deflected away from obstacles surrounding it during movement. Hence, the direction of motion θ_{md} obtained above is adjusted to avoid the risk of collision with these obstacles. In order to solve this issue, we propose an algorithm which is an evolution of the one introduced in the SND. As compared with the SND, the key difference in our approach is that it generates safer paths and avoids deadlocks which occur in some cases without affecting the smoothness property as shown later on. The proposed solution is described in the following.

Each obstacle from among N obstacles falling within the safe distance D_{s} around the robot imposes a threat (t_i) dependent on the proximity of this obstacle to the robot boundary [19].

$$t_{\mathrm{i}} = \mathrm{sat}_{[0,1]}\left(\frac{D_{\mathrm{s}}-D_{\mathrm{i}}}{D_{\mathrm{s}}}\right) \tag{10}$$

where D_{i} is the distance to the i^{th} obstacle point measured from the robot boundary. The value of the threat is 0 when the obstacle is outside D_{s} and 1 if the robot touches the obstacle.

Dependent on the threat calculated for each obstacle, a deflection from the direction of motion θ_{md} will be applied to avoid each of these obstacles [19].

$$\delta_{\mathrm{i}} = t_{\mathrm{i}} \,.\, \mathrm{proj}(\mathrm{dist}_{\mathrm{cc}}(\theta_{\mathrm{i}} + \pi, \theta_{\mathrm{md}})) \in [-\pi, \pi[\tag{11}$$

where θ_{i} is the angle towards the i^{th} obstacle point and proj $(\mathrm{dist}_{\mathrm{cc}}(\theta_{\mathrm{i}} + \pi, \theta_{\mathrm{md}}))$ is the position of θ_{md} measured counter-clockwise from the angle opposite the obstacle i. This value is multiplied by t_{i} to make the deflection dependent on the proximity of the obstacle to the robot.

In the SND approach, all threats on the two sides of the robot falling within the safe distance are considered while calculating the total weighted deflection. If one side has a large number of obstacles (threats) compared with the other, a high deflection will be applied towards the side with fewer threats. This enforces the robot to hit obstacles on that side if the gap is narrow. The problem increases when D_{s} is enlarged since it will cover more area containing more obstacles and so the difference between the two sides increases. This drawback does not exist in the ND+ approach since it considers only the closest obstacle points on the left and right of the robot. However, deflecting the direction of motion from the closest obstacle causes sharp changes in the trajectory of the robot which reduces smoothness. This problem can be solved by considering the ratio of threat numbers on the two sides as shown hereinafter. Another drawback increasing the problem stems from the fact that the weight $({t_{\mathrm{i}}}^2)$ used in SND is not strictly for close obstacles since it is between 0 and 1. Its square does not differentiate strongly between close and farther obstacles relative to the robot. As the safe distance increases, the threat difference between two obstacles on different positions decreases causing a promotion of the problem.

Our proposed solution extends the difference between threats and behaves stricter when an obstacle gets closer to the robot. This is achieved through modifying the function calculating the weight as follows:

$$w_{\mathrm{i}} = \frac{1}{(1-t_{\mathrm{i}})^k} \tag{12}$$

where k defines the strength of the weight. Increasing the value of k ensures safer behavior through moving away from the closest obstacles (it must not be high since it decreases smoothness). From our tests k should be searched for in the range [1, 3]. As t_{i} gets closer to 1, the output increases more sharply. This is required to ensure a stricter deflection from the closest obstacles. In this equation t_{i} should not equal 1.

After that, we divide the space into two regions, one to the right and the other to the left of the robot's direction of motion. The total weighted deflection is taken for each side separately as shown below.

Suppose that N_{L} and N_{R} are the number of obstacles inside D_{s} on the left and right sides. We can define the total weights for all obstacles on the left side as:

$$W_{\mathrm{L}} = \sum_{\mathrm{i}=1}^{N_{\mathrm{L}}} w_{\mathrm{i}} \tag{13}$$

The total deflections on the left side can now be defined as the weighted sum of all obstacle deflections on this side.

$$D_{\mathrm{L}} = \sum_{i=1}^{N_L} \frac{w_{\mathrm{i}}}{W_{\mathrm{L}}} \delta_{\mathrm{i}} \in [-\pi, \pi[\tag{14}$$

The value of D_{L} is changed to adjust the difference in the number of obstacles inside D_{s}, between the two sides.

$$D_{\mathrm{L}} = D_{\mathrm{L}}/P_{\mathrm{L}} \tag{15}$$

where $P_{\mathrm{L}} = N_{\mathrm{L}}/N$. Note that we set $P_{\mathrm{L}} = 1$ if either N_{L} or N equals zero.

We find $\mathrm{W_R}$ and D_{R} for the right hand side in the same manner as (Eqs. 13, 14 and 15).

Finally, the total net deflection D_{net} is calculated as follows:

$$D_{\mathrm{net}} = \frac{W_{\mathrm{L}} . D_{\mathrm{L}} + W_{\mathrm{R}} . D_{\mathrm{R}}}{(W_{\mathrm{L}} + W_{\mathrm{R}})} \tag{16}$$

In order to achieve safety in navigation, the angular trajectory for the robot is the direction of motion θ_{md} modified by the net deflection D_{net}.

$$\theta_{\mathrm{traj}} = \theta_{\mathrm{md}} - D_{\mathrm{net}}. \tag{17}$$

The speed of the robot is controlled according to the distance between the robot and the closest obstacle. Suppose that d_{min} is the distance between the closest obstacle and the robot boundary. The maximum speed of the robot should be limited as follows:

$$v_{\mathrm{limit}} = \mathrm{sqrt}\left(1 - \mathrm{sat}_{[0,1]}\left(\frac{D_{\mathrm{vs}}-d_{\mathrm{min}}}{D_{\mathrm{vs}}}\right)\right) . v_{\mathrm{max}} \tag{18}$$

where v_{max} is the maximum linear speed of the robot and D_{vs} the velocity safe distance.

Remark 2 (Comparison to SND): The safe distance used in Eq. (18) to limit the speed D_{vs} is different from the one used above to calculate the threats D_{s}. In this regard, we can increase D_{s} to be more reactive for dynamic obstacles without affecting the speed. In the SND, this is not possible for two reasons: enlarging D_{s} will increase the possibility of the deadlock mentioned above and will decrease the speed (may be too slow to move the robot) in cluttered environments. Furthermore, the nonlinear function ($sqrt$) is used to increase the speed of the robot as compared with the linear equation defined in the SND.

In order to calculate the linear and angular speeds, we use the same equations proposed in [16] and [19].

$$v = \text{sat}_{[0,1]}\left(\frac{\pi/_4 - |\theta_{\text{traj}}|}{\pi/_4}\right) \cdot v_{\text{limit}} \qquad (19)$$

$$w = \text{sat}_{[-1,1]}\left(\frac{\theta_{\text{traj}}}{\pi/_2}\right) \cdot w_{\max} \qquad (20)$$

where $w_{\max}$ is the maximum orientation (angular) speed of the robot.

IV. SIMULATIONS

In order to explain the advantages of our work as compared with the SND method, we implemented the two approaches in the well known Player/Stage robot software system version 2.1.1 using the same specifications for the laser rangefinder used in our experiments. This sensor scans 683 points over 240° with a maximum range of 5.6 *m*. We used a rectangular differential drive robot with a length of 0.53 *m* and a width of 0.49 *m* in order to imitate the real robot. We limited the maximum linear and angular velocities to 0.5 *m/s* and 1.0 *rad/s* while the safe distance was set to 1.0 *m*.

We show four different scenarios applied on various created maps in order to clarify the importance of our approach. The first two simple scenarios show the advantages of our new method for analyzing gaps compared with the ND and SND. The other two scenarios demonstrate the power of our approach in avoiding the deadlock problem mentioned in section III-D.

A. Simulations for Scenarios (1, 2)

This part explains how fetching gaps in the CG alleviates oscillations and achieves safe and smooth trajectory. Scenario (1) is shown in Figs. 4(a-d), where the task is to move the robot from the start to the goal locations marked in the map. Using the SND method, the robot reaches an oscillation behavior and stops at the point shown in Fig. 4a. This is due to the fact that the navigable region closest to the goal is formed by discontinuity points *A* and *B* (the other regions occur at points: *D*, *E* and *F*, *G*). When the robot starts moving towards the rising gap of this region (*B*), another discontinuity appears on point *C* which adds a new region: *C*, *D* (see Fig. 4b). Now, this region will be the closest one to the goal which forces the robot to navigate towards its rising gap (*C*). The robot will alternate between these two states without reaching the goal. The gaps detected using the CG method are marked by arrows 1, 2 and 3 in Fig. 4c. In all the cases, the discontinuity point *E* forms gap 2 which is the closest one to the goal. This forces the robot to navigate through it avoiding all obstacles till reaching the goal as shown in Fig. 4d.

The other scenario (2) demonstrates the advantage of adding the angle α to the motion direction θ_{md} as stated in section III-C. This scenario implies moving the robot from its start position to the goal through the unique opening as shown in Figs. 4e, f. The region identified by the SND method (from point *A* to *B*) is too narrow as seen by the robot (Fig. 4e). The robot moves to the center of the region which is nearly towards the obstacles identified by point *P*, and then it will be deflected when it gets close to these obstacles. Adjusting θ_{md} to fit the robot diameter in the CG approach forces the robot to navigate far from the obstacles on point P as shown in Fig. 4f. Hence, a safer and smoother trajectory is achieved.

B. Simulations for Scenarios (3, 4)

These two scenarios explain the deficiency of the SND method when there is a large difference in the number of threats on the two sides of the robot. The first one shows a situation where the robot should pass two narrow openings in order to reach its goal. By using the SND method, the robot collides with obstacles on the side (*S*1) of the first opening, coming to a full stop as shown in Fig. 4g. This is because side (*S*2) has more obstacles (a high deflection) than side (*S*1). The CG method overcomes this problem allowing a safe and smooth navigation when passing the two openings towards the specified goal (see Fig. 4h).

The route chosen for scenario (4) contains tight corridors, as explained in Figs. 4i, j, where the objective is to pass them safely and smoothly. The robot moved very close to obstacles on the corridors labeled *A*, *B* and *C* when we implemented the SND algorithm as shown in Fig. 4i. This refers to the same problem of the high deflection towards the side containing fewer obstacles mentioned previously. Adjusting the difference between threats on the two sides and providing stricter deflection from the nearest obstacles solve this drawback as stated in section III-D. This can be noticed in the CG behavior shown in Fig. 4j where the robot navigates from the mid of the tight corridors. No actual differences in smoothness are noticed between the SND and the CG methods except on some locations where the CG did better. This can be sensed from the points labeled 1, 2 and 3 on their corresponding figures. We support that by plotting the angular velocity w against time for the SND and the CG methods as shown in Fig. 4k-1 and Fig. 4k-2. Moreover, the goal was reached in 140 *sec* using the SND method while it took 125 *sec* in the case of the CG.

V. EXPERIMENTAL RESULTS

We have confirmed the simulated results using our real robot, a differential drive Pioneer 3-AT mobile robot equipped with a Hokuyo URG-04LX laser scanner and an on-board 2 GHz Pentium M computer. The robot platform is rectangular (0.53 × 0.49 *meters*) with non-holonomic constraints. The maximum translational velocity of the robot is 0.7 *m/s* and the maximum rotational one is 2.4 *rad/s*. In our experiments we limited these velocities to 0.5 *m/s*, 1.0 *rad/s*.

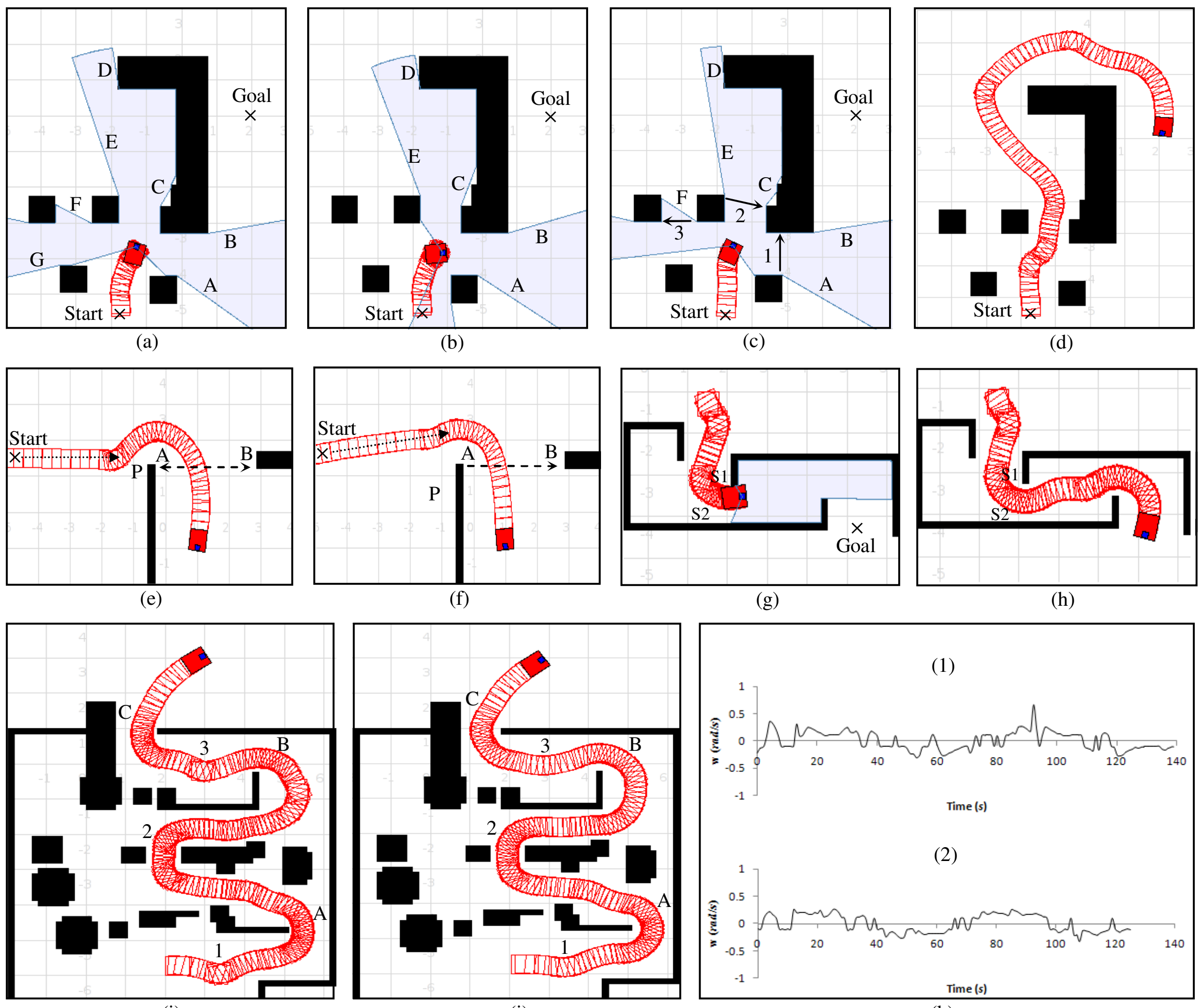

Fig. 4. (a and b) Oscillation in behavior using the SND method. (c and d) No oscillation occurs in CG. (e) Trajectory followed with the SND algorithm, where the gap aperture with respect to the robot vision is small. (f) Trajectory followed by the CG method, where the gap aperture with respect to the robot vision is small. (g) Deadlock occurs in SND, with numerous threats on one side and fewer threats on the other. (h) Safe path with the CG method, with numerous threats on one side and fewer threats on the other. (i) Trajectory followed by the SND. (j) Trajectory followed with the CG. (k) Angular speed versus time.

We adapted the algorithm introduced in this paper to accommodate the rectangular shape of our robot. Furthermore, the inaccuracy of sensor readings is alleviated by using a median filter. A detailed description concerning these two issues will be presented in a future paper.

Fig. 5a shows one of our experiments. The only information provided to the robot in advance was the goal location. The experiment was carried out using SND and CG. While travelling through the first openings, e.g. passage 1, the two methods behaved fairly similar but differences become clearer when looking at passage 2. Using the SND algorithm, the robot moved close to side 1 of the opening (Fig. 5b) and came to a full stop while nearly touching the obstacle (Fig. 5c). Using our new CG method the robot safely moved through this passage and reached the goal (Fig. 5d,e). The recorded angular speeds are shown in Fig. 5f-1 and Fig. 5f-2.

VI. Conclusions And Future Works

We have addressed the Closest Gap Navigation (CG) method for local reactive collision avoidance. CG alleviates oscillations and computational complexity by designing a new scheme for fetching gaps which reduces their number and eliminates unnecessary ones. The robot vision of the opening angle of the gap is taken into account also in order to provide a smoother behavior. Moreover, it improves the safety of paths generated by the Smooth Nearness-Diagram (SND) method through considering the ratio of threats on the two sides of the robot and applying stricter deviation against an obstacle as it gets closer to the robot. As a consequence, a robust navigation in very dense and cluttered scenarios is achieved.

Future work on the CG includes finding an analytical fully proofed solution that ensures safety in all cases and a random distribution of obstacles. Also, an optimal speed will be calculated taking into consideration the kinematics and dynamics of the robot.

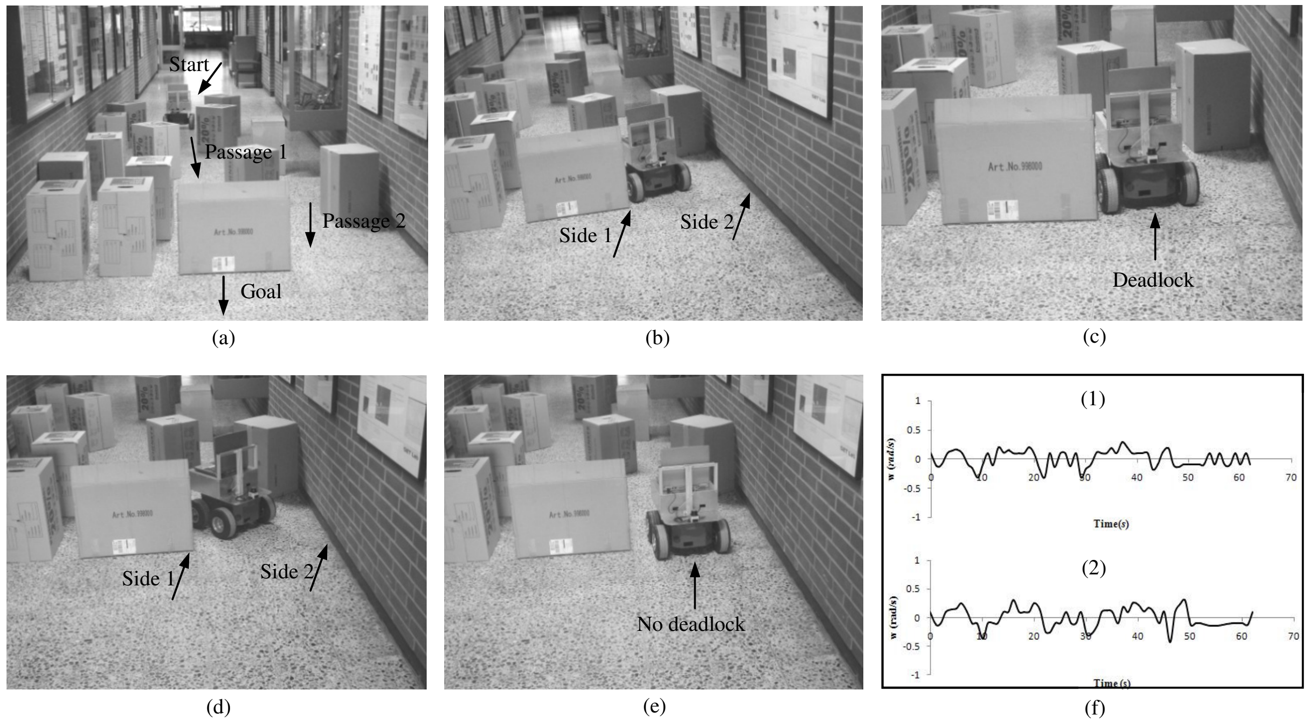

Fig. 5. (a) Experimental setup. (b) The robot moves close to side 1 using the SND method. (c) A deadlock occurs with the SND method, with numerous threats on one side and fewer threats on the other. (d) The robot navigates safely using the CG method. (e) Passing a gap with the CG method, with numerous threats on one side and fewer threats on the other. (f) Angular speed versus time for the SND (1) and the CG methods (2).

REFERENCES

[1] P. Cao, X. Liao, and E. L. Hall, "Motion Control Design of the Bearcat II Mobile Robot," SPIE, Intelligent Robots and Computer Vision XVIII: Algorithms, Techniques, and Active Vision, vol. 3837, pp. 118-126, 1999.

[2] A. Sanchez, X. Hernandez, O. Torres and Alfredo Toriz P., "Mobile robots navigation in industrial environments", 2009 Mexican International Conference on Computer Science, pp.155-166, 2009.

[3] S.Hirose and E. Fukushima, "Snakes and Strings: New Robotic Components for Rescue Operations," Springer Tracts in Advanced Robotics, Experimental Robotics VIII, volume 5, pp. 48-61, 2003.

[4] "Robots - our helpers in space", *European Space Agency*, http://www.esa.int/esaCP/SEM8XXWJD1E_index_0.html, 29 November 2004.

[5] O. Khatib, "Real-time Obstacle Avoidance for Manipulators and Mobile Robots", *The Intl. Journal of Robotics Research*, 5(1), pp.90- 98,1986.

[6]R. C. Arkin, Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research*, pp. 92-112*,* August 1989.

[7] R. B. Tilove, "Local obstacle avoidance for mobile robots based on the method of artificial potentials," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 2, Cincinnati, OH, pp. 566–571, 1990.

[8] M. Khatib, "Sensor-based motion control for mobile robots," Ph.D. dissertation, LAAS-CNRS, Toulouse, France, 1996.

[9] M. Khatib and R. Chatila, "An Extended Potential Field Approach for Mobile Robot Sensor-based Motions", *In Intl. Conf. On Intelligent Autonomous Systems IAS'4*, 1995.

[10] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 1179–1187, May 1989.

[11] Moravec, H.P., and Elfes, A., "High resolution maps from wide angle sonar", *IEEE Int. Conf. on Robotics and Automation*, pp. 116- 121, 1985.

[12] Y. Koren and J. Borenstein, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation", In *IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 1398–1404, Sacramento, 1991.

[13] J. Borenstein and Y. Koren, "The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots", *IEEE Transactions on Robotics and Automation*, pp.278-288, 7(3), June 1991.

[14] I. Ulrich and J. Borenstein, "VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots", *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1572-1577, Leuven, Belgium, May 1998.

[15] I. Ulrich and J. Borenstein, "VFH*: Local Obstacle Avoidance with Look-Ahead Verification." *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2505-2511, San Francisco, California, Apr. 2000.

[16] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, Vol. 20, Issue 1, pp. 45-59, Feb. 2004.

[17] R. C. Arkin, *Behavior-Based Robotics*. Cambridge, MA: MIT Press, 1999.

[18] J. Minguez, J. Osuna, and L. Montano, "A "divide and conquer" strategy based on situations to achieve reactive collision avoidance in troublesome scenarios," in *IEEE Int. Conf. on Robotics and Automation*, (New Orleans, LA), pp. 3855–3862, Apr. 2004.

[19] Joseph W. Durham and Francesco Bullo, "Smooth Nearness-Diagram Navigation," *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, pp. 690-695, 22-26 Sept. 2008.

[20] Y. Lin, C. Chou and F. Lian, "Indoor Robot Navigation Based on DWA*: Velocity Space Approach with Region Analysis," in *ICROS-SICE International Joint Conference 2009*, pp. 700-705, 18-21 Aug. 2009.

[21] R. Simmons, "The Curvature-Velocity Method for Local Obstacle Avoidance", *In IEEE Intl. Conf. on Robotics and Automation ICRA'96,* pp.2275-2282,Minneapolis, April 1996.

[22] D. Fox, W. Burgard. and S. Thrun, "The Dynamic Window Approach to Collision Avoidance*", IEEE Robotics and Automation Magazine*, 4(1), pp.23-33, March 1997.

[23] D. Fox, W. Burgard., S. Thrun and A.Cremers, "A Hybrid Collision Avoidance Method for Mobile Robots", *in IEEE Intl. Conf. On Robotics and Automation ICRA'98*, pp.1238- 1243, 1998.

[24] O. Brock and O. Khatib, "High-Speed Navigation Using the Global Win-dow Approach", *in IEEE Intl. Conf. on Robotics and Automation ICRA'99*, Detroid, Michigan, pp.341-346, May 1999.

[25] Graham, Stephen, "America's robot army", *New Statesman*, http://www.newstatesman.com/200606120018, 12 June 2006.

[26] "Battlefield Robots: to Iraq, and Beyond", *Defense Industry Daily*, http://www.defenseindustrydaily.com/battlefield-robots-to-iraq-and-beyond-0727, 20 June 2005.

[27] Steven M. LaValle, "Planning Algorithms," Cambridge University Press, 2006.