IROS 2008, To appear
Nice, France

Smooth Nearness-Diagram Navigation

Joseph W. Durham and Francesco Bullo

***Abstract*— This paper presents a new method for reactive collision avoidance for mobile robots in complex and cluttered environments. Our technique is to adapt the "divide and conquer" approach of the Nearness-Diagram+ Navigation (ND+) method to generate a single motion law which applies for all navigational situations. The resulting local path planner considers all the visible obstacles surrounding the robot, not just the closest two. With these changes our new navigation method generates smoother motion while avoiding obstacles. Results from comparisons with ND+ are presented as are experiments using Erratic mobile robots.**

I. INTRODUCTION

Safe navigation through an environment is a fundamental piece of most potential tasks for autonomous mobile robots. Autonomous robots are being developed for search-and-rescue [1], transportation [2], and mobility assistance [3], among many other applications. In each circumstance, the safety and performance of navigation in unknown and dynamic environments with a potential high density of obstacles is crucial to accomplishing the larger task.

One intriguing concept which has recently shown a lot of potential in mobile applications is *gaps*: discontinuities in the depth of obstacles around the robot which indicate potential paths into occluded areas of the environment. Navigation and exploration based solely on gaps, as opposed to the more common occupancy grid maps, has been studied in [4]. [4] introduced the Gap Navigation Tree (GNT) which contains links of which gaps lead to which other gaps. Navigation based on GNTs was shown to be intrinsically distance-optimal without any need for distance measurements or localization [5].

The Nearness-Diagram Navigation (ND) method [6] was the first reactive navigation approach based on gaps. By navigating based on gaps, ND avoids local trap situations without the computational load of determining which areas of the environment are connected. The ND method uses a tree of conditions based on the configuration of the obstacles closest to the robot to divide navigation behavior into five scenarios. The subsequent ND+ method [3] adds a sixth scenario to balance the division of motion laws and increases the smoothness of transitions between some of the scenarios. We describe the six ND+ scenarios in more detail after introducing some required concepts in Section III.

This work was supported in part by NSF Awards IIS-0330008 and CMS-0626457.

J. W. Durham and F. Bullo are with the Department of Mechanical Engineering, University of California, Santa Barbara, CA 93106 joey@engineering.ucsb.edu, bullo@engineering.ucsb.edu

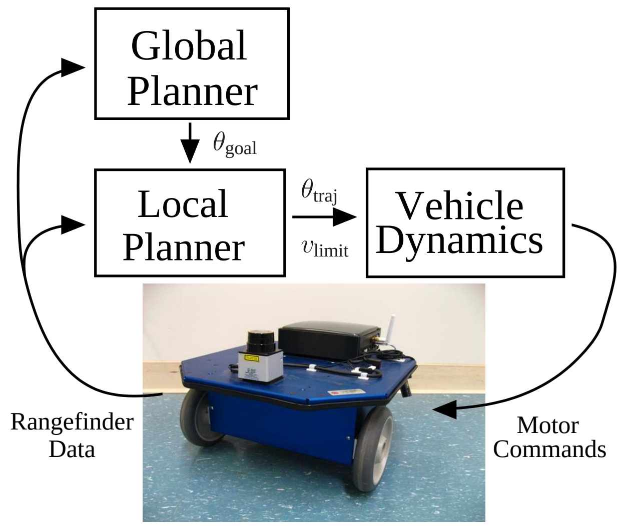

Fig. 1. The motion control framework considered by this paper. The proposed SND method fills the role of the local planner.

In this paper we present the Smooth Nearness-Diagram Navigation (SND) method that is an evolution of ND+. As compared with the ND+ navigation scheme [3], the key difference in our approach is that a single motion law is proposed that is applicable to all possible configurations of surrounding obstacles. The change away from separate motion laws for different scenarios, as we will describe in Section III, removes abrupt transitions in behavior when the robot navigates near obstacles. In addition, adjusting the gap trajectory based on all nearby obstacles, not just the closest two, leads to smoother paths as we will show in Section IV.

II. MOTION CONTROL FRAMEWORK

The focus of this paper is on the reactive collision avoidance (local planner) component of the robot motion control framework shown in Fig. 1. The distinction between the role of the global and local planners is fundamentally important to this motion control framework and the measurement of success for the two planners. Similar task separation schemes for motion control have been considered in [7], [2] among others.

In this framework, the robot is equipped with sensors capable of producing a 2-D depth map of obstacles surrounding the robot. The most common forms of such sensors are sonar and laser range finders. Each sensor update passes obstacle location information to both the global and local planners.

The global planner is responsible for keeping track of the relative position of the robot and its goal. The global planner must also remember which potential paths towards the goal have been determined to be dead-ends. Using this information, it passes a desired heading θ_{goal} to the local planner though it is not necessary to update this heading at every sensor update. Examples of global planners which could fill this role include GNTs [4] and D* [8], among others.

The local planner considers the global goal heading and the local obstacles visible to the robot to plot a trajectory which will make safe progress towards the goal. This reactive planner must be able to process and react to each sensor update to successfully avoid obstacles. Examples of local planners include Nearness-Diagram methods, VFH [9], and Dynamic Window [10]. The local planner passes a trajectory θ_{traj} and a speed limit v_{limit} to the robot dynamics component which translates the desired trajectory into commands for the various actuators of the robot.

III. Reactive Collision Avoidance Method

In this section we present the Smooth Nearness-Diagram Navigation method (SND) for collision avoidance which fills the role of the local, reactive planner in the motion control framework in Section II.

The SND method works as follows: first, the rangefinder data is analyzed to determine the structure of obstacles surrounding the robot as described in Section III-B. The best heading which makes progress towards the goal direction set by the global planner is then selected as presented in Section III-D. In Section III-E we describe how the SND method deflects this heading to avoid any nearby obstacles. This process of determining a safe trajectory is repeated for each sensor update.

A. Definitions

Angles and angular distances play a significant part in our algorithm and so we will carefully define these relationships. Let $\mathbb{S}^1$ be the unit circle attached to the robot's reference frame. We will measure positions on $\mathbb{S}^1$ counterclockwise from the positive horizontal axis (directly in front of the robot). Positive angles less than π are on the left of the robot, negative on the right.

For angles $\alpha, \beta \in \mathbb{S}^1$, we let $\text{dist}(\alpha,\beta)$ be the geodesic distance between α and β defined by $\text{dist}(\alpha,\beta) = \min\{\text{dist}_{\mathsf{c}}(\alpha,\beta), \text{dist}_{\mathsf{cc}}(\alpha,\beta)\}$, where $\text{dist}_{\mathsf{c}}(\alpha,\beta) = ((\alpha - \beta) \bmod 2\pi)$ and $\text{dist}_{\mathsf{cc}}(\alpha,\beta) = ((\beta - \alpha) \bmod 2\pi)$ are the path lengths from α to β traveling clockwise and counter-clockwise, respectively. Here $(\alpha \bmod 2\pi)$ is the remainder of the division of α by 2π.

Given a scalar θ, we let $\text{proj}(\theta)$ take value in $[-\pi, \pi[$, where the map $\text{proj} : \mathbb{R} \to [-\pi, \pi[$ is defined by

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

Given $a < b$, we define the saturation function $\text{sat}_{[a,b]} : \mathbb{R} \to [a,b]$ by

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

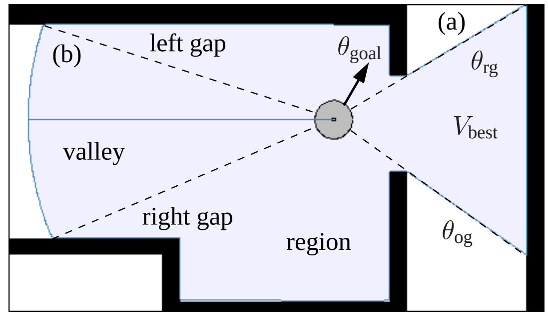

Fig. 2. The circular robot detects four gaps, indicated by dashed lines, in the depth measurements around it. There are two types of gaps: (a) is created by neighboring depth measurements differing by more than the robot diameter while at (b) one depth measurement returns no obstacle in range. The light solid line indicates the angle directly behind the robot. The four gaps define regions and valleys around the robot, some of which are labeled. The robot choose V_{best}, θ_{rg}, and θ_{og} based on the goal direction θ_{goal}.

B. Locating Gaps and Valleys

A *gap* occurs at an angle where two contiguous depth measurements are either separated by more than the robot diameter $2R$ or one of the measurements returns no obstacle in range. The first type of gap occurs at (a) in Fig. 2, the second at (b). See [6] for more on calculating the location of gaps from rangefinder data. A "*left gap*" means that the closer measured obstacle falls on the left side of the gap, as in (a) in Fig. 2, indicating that there may be an occluded free area on the left. The opposite holds for right gaps.

Each pair of consecutive gaps defines a *region*. A *valley* is a navigable region with either a left gap on its left side, a right gap on its right side, or both. Each valley is defined by two gaps, one of which we will call the *rising gap* and refer to by the angle θ_{rg}. When a valley is defined by two left or right gaps, the gap on the appropriate side of the valley is θ_{rg}. If the valley is defined by a left and a right gap (as is the case in Fig. 2), then the gap closest to the heading provided by the global planner θ_{goal} is selected as θ_{rg}. The *other gap* of the valley is referred to as θ_{og}.

Once the list of valleys surrounding the robot is assembled, each θ_{rg} is compared against θ_{goal}. The valley which best matches this heading, V_{best}, is selected as shown in Fig. 2. The mechanism for selecting the best valley are the same as in the ND and ND+ methods, the differences come with the selection of a desired heading and adjustments for nearby obstacles.

C. ND+ Method Description

The ND+ method divides behavior into six scenarios based on the obstacle point closest to the left and right side of the robot [3]. The top classifier, low or high safety, depends on whether an obstacle falls within the safety distance D_s:

1) HSGR: High safety, θ_{goal} is in V_{best};
2) HSWR: High safety, V_{best} is wide;
3) HSNR: High safety, V_{best} is narrow;
4) LSGR: Low safety, θ_{goal} is in V_{best};
5) LS1: Low safety, close obstacle on only one side;
6) LS2: Low safety, close obstacles on both sides.

The ND+ method determines the desired trajectory for the robot by deflecting θ_{rg} based on the two closest obstacles and the width of V_{best}.

D. Determining Desired Heading

The two gaps of V_{best}, θ_{rg} and θ_{og}, define the *free walking area* for the robot which makes the best progress towards the goal. We will now define two angles based on this valley, first the safe rising gap θ_{srg}:

$$\theta_{\text{srg}} = \begin{cases} \theta_{\text{rg}} - \arcsin\left(\frac{R+D_{\text{s}}}{D_{\text{rg}}}\right), & \text{if } \theta_{\text{rg}} \text{ is a left gap,} \\ \theta_{\text{rg}} + \arcsin\left(\frac{R+D_{\text{s}}}{D_{\text{rg}}}\right), & \text{if } \theta_{\text{rg}} \text{ is a right gap} \end{cases} \tag{3}$$

where θ_{rg} and D_{rg} are the angle of the rising gap and the distance to the obstacle creating the gap from the center of the robot. This adjustment to θ_{rg} will point the robot in such a way that the obstacle creating the gap will not enter D_{s} as the robot moves towards the gap. When V_{best} is narrow, it is possible that θ_{srg} will point too close to θ_{og}. For these narrow valleys it is better to head towards the angle which bisects V_{best}, θ_{mid} defined by:

$$\theta_{\text{mid}} = \begin{cases} \theta_{\text{rg}} - \text{dist}_{\mathsf{c}}(\theta_{\text{rg}}, \theta_{\text{og}})/2, & \text{if } \theta_{\text{rg}} \text{ is a left gap,} \\ \theta_{\text{rg}} + \text{dist}_{\mathsf{cc}}(\theta_{\text{rg}}, \theta_{\text{og}})/2, & \text{if } \theta_{\text{rg}} \text{ is a right gap,} \end{cases} \tag{4}$$

where the half-width of V_{best} is subtracted from θ_{rg} for left gaps and added for right gaps.

Under most circumstances, the desired heading for the robot θ_{d} will be whichever of θ_{srg} and θ_{mid} is closer to θ_{rg}:

$$\theta_{\text{d}} = \begin{cases} \theta_{\text{mid}}, & \text{if } \text{dist}(\theta_{\text{d}}, \theta_{\text{mid}}) < \text{dist}(\theta_{\text{d}}, \theta_{\text{srg}}), \\ \theta_{\text{srg}}, & \text{else.} \end{cases} \tag{5}$$

Remark 1 (Comparison to ND+): Both θ_{srg} and θ_{mid} are all also used by some of the cases in the ND+ method. The key difference is that our approach chooses whichever is closer to θ_{rg} in all scenarios, removing a source of non-smoothness in some of the transitions between the cases in ND+ (particularly LS1 and LS2). •

It is also worth noting that if the goal of the robot is to assume a particular position, then θ_{d} should be set to θ_{goal} when θ_{goal} falls between θ_{rg} and θ_{og}. We consider this to be a special case as moving through the environment safely is the primary goal of the reactive planner. In addition, some visibility-based tasks can be accomplished with distance-optimal paths simply by chasing gaps [5].

E. Obstacle Avoidance Method

With the desired heading θ_{d} determined, the SND method will consider deflecting this trajectory based on the configuration of obstacles surrounding the robot. In [3], the ND+ method separated the actions the robot would take into six different scenarios based on the proximity of obstacles on the left and right of the robot, the width of V_{best}, and θ_{goal}. Our approach is to generate a single obstacle avoidance rule which works under all scenarios and considers all of the obstacles around the robot, not just the closest two.

The foundation of the SND method is the measurement of the threat posed by each of the N obstacle distance measurements from the rangefinder. An obstacle is considered a threat if it falls within the safety distance D_{s} of the boundary of the robot and the threat measure s_{i} increases as the obstacle gets closer to the robot.

$$s_{\text{i}} = \text{sat}_{[0,1]}\left(\frac{D_{\text{s}} + R - D_{\text{i}}}{D_{\text{s}}}\right) \tag{6}$$

where D_{i} is the distance to the i^{th} obstacle point measured from the center of the robot and the sat operator caps s_{i} at 0 when the obstacle is outside D_{s} and 1 if the robot is touching the obstacle.

Using this measurement of the danger posed by each visible obstacle we can define the deflection from the desired heading to avoid each of these obstacles, δ_i.

$$\delta_i = s_{\text{i}} \cdot \text{proj}\left(\text{dist}_{\mathsf{cc}}\left(\left(\theta_i + \pi\right), \theta_{\text{d}}\right)\right) \in [-\pi, \pi[\tag{7}$$

where θ_i is the angle towards the i^{th} obstacle point and the term $\text{proj}\left(\text{dist}_{\mathsf{cc}}\left(\left(\theta_i + \pi\right), \theta_{\text{d}}\right)\right)$ is the position of θ_{d} measured counter-clockwise from the angle directly away from the obstacle. This angular distance is weighted by s_{i}: when s_{i} is 0 and the obstacle is outside D_{s}, the deflection δ_i is also 0. When the robot is touching the i^{th} obstacle and s_{i} is 1, δ_i is at full strength and will point directly away from the obstacle regardless of the value of θ_{d}.

To define the relative importance of each δ_i we use the sum of the square of all the s_{i} danger coefficients:

$$s_{\text{total}} = \sum_{i=1}^{N} s_{\text{i}}^2. \tag{8}$$

With this we can now define the total obstacle avoidance deflection Δ_{avoid} as the weighted sum of all δ_i:

$$\Delta_{\text{avoid}} = \sum_{i=1}^{N} \frac{s_{\text{i}}^2}{s_{\text{total}}} \delta_i \in [-\pi, \pi[. \tag{9}$$

When there is a single obstacle point inside D_{s}, the effect of Eq. (9) is equivalent to the obstacle avoidance deflection in the LS1 (close obstacle on one side of the robot) condition from [3]. However, when there are multiple obstacle points inside D_{s} (either from multiple obstacles or large obstacles), Eq. (9) accounts for all of them and finds the weighted net avoidance deflection. Terms for which s_{i} is larger will have more pull in the sum, as will obstacles closer to θ_{d} because of the differencing in Eq. (7).

The safe angular trajectory for the robot is then the goal directed angle θ_{d} adjusted by the obstacle avoidance deflection Δ_{avoid}:

$$\theta_{\text{traj}} = \theta_{\text{d}} + \Delta_{\text{avoid}}. \tag{10}$$

Note that since δ_i is formulated as a deflection away from θ_{d}, if the robot is very close to an obstacle, then θ_{traj} may point in nearly the opposite direction as θ_{d}. When the robot moves away from the obstacle, Δ_{avoid} will shrink and the robot will follow θ_{d}. There is also no hard constraint against moving towards one obstacle (particularly one outside D_{s}) in order to avoid another.

Equation (10) determines the new heading for the robot. Our obstacle avoidance layer also specifies the speed limit v_{limit} of the robot to maintain safety near obstacles.

$$v_{\text{limit}} = (1 - \min\{s_1, ..., s_N\}) \cdot v_{\max} \tag{11}$$

where $v_{\max}$ is the maximum velocity of the robot. The robot slows down based upon the closest obstacle, coming to a full stop if it ever touches an obstacle.

Remark 2 (Smoothness Properties of SND): Throughout Section III we have argued that by using a single motion law in all circumstances and by taking all nearby obstacles into account, SND produces smoother motion than ND+. We show simulations confirming this in Section IV. Beyond these arguments, we conjecture that the continuous version of Eq. (10) is continuously dependent on the position of the robot. Let us provide some arguments to support this conjecture. For a rangefinder with infinitesimal angular resolution, Eq. (8) becomes:

$$s_{\text{total}}(x, y) = \oint s(\alpha, x, y)^2 d\alpha. \tag{12}$$

where $s(\alpha, x, y)$ is the continuous version of s_{i} from Eq. (6). Since it includes s is dependent linearly on the visibility distance, Eq. (12) is reminiscent of the formula for the area of the visibility space of the robot:

$$A_{visible}(x, y) = \oint r(\alpha, x, y)^2 d\alpha. \tag{13}$$

For a polygonal environment, possibly non-convex and with holes, the area of the visibility space is locally Lipschitz continuous everywhere except at the internal reflex vertices of the environment (an impossible position for a robot of non-zero size) [11]. In future work these observations could potentially be extended to prove that the continuous version of Δ_{avoid} is continuously dependent on the position of the robot. •

IV. SIMULATIONS

To demonstrate the differences in the execution of the SND method and the ND+ method presented in [3], we implemented both in version 2.0.3 of the open-source Player/Stage robot software system [12]. One of the strengths of Player/Stage is that the same code can run either a real or simulated robot. We will show results from SND running on a physical robot in Section V, but simulations allow the two methods to be directly compared with no differences other than their actions. For the simulations we used a raytrace accuracy of $0.02m$.

Both SND and ND+ are designed to handle troublesome scenarios with very close obstacles. For these simulations we created a map with many tight squeezes between obstacles where the robot could pass with less then $10cm$ total clearance. The map can be seen in Fig. 3, where the black regions are obstacles. Instead of using a global planner for these simulations the robot is instructed to follow gaps towards the top right corner of the map and then towards the top left. The robot's goal is to move through the environment in this way, not to assume a particular position.

A. Robot Model

For simplicity we have used a circular, differential-drive robot with $R = 0.25m$ and a weight of $12kg$. The simulated laser rangefinder samples $n = 1024$ points over a full $360°$ with a range of $4m$. The linear and angular velocities are capped at $v_{\max} = 0.5m/s$ and $\omega_{\max} = 1.0rad/s$ while the safety boundary around the robot is set to $D_{\text{s}} = 1.5R = 0.375m$.

A differential-drive robot in Player/Stage accepts two motion commands: rotational and linear speeds. These speeds are calculated from θ_{traj} and v_{limit} using the following equations:

$$\omega = \text{sat}_{[-1,1]}\left(\frac{\theta_{\text{traj}}}{\pi/2}\right) \cdot \omega_{\max}, \tag{14}$$

$$v = \text{sat}_{[0,1]}\left(\frac{\pi/4 - |\theta_{\text{traj}}|}{\pi/4}\right) \cdot v_{\text{limit}}. \tag{15}$$

If θ_{traj} points in the opposite direction as the current robot heading, the robot will first spin in place for $\frac{3\pi}{4}$. Once its heading is within $\frac{\pi}{4}$ of θ_{traj} the robot will begin moving forward with a velocity proportional with its alignment to θ_{traj}. Equations (14) and (15) are similar to the those used in [6].

B. SND Simulation

The route chosen using the SND method is shown in Fig. 3. The robot successfully navigates the course in $135sec$, slowing down to squeeze between tight obstacles. At regular intervals a square bounding box is left behind on the map indicating the progress of the robot. The relative speeds for different sections of the map can be interpreted from the density of these gray trails. Denser sections also correspond to the parts of the path where the robot passes close to obstacles, since v_{limit} is determined by the closest obstacle in Eq. (11).

C. ND+ Simulation

The route chosen using the ND+ method from [3] is shown in Fig. 4. The robot does not complete the course after clipping the last obstacle, coming to a full stop after $254sec$. Other close brushes with obstacles can be seen along the path taken, particularly between B and C. In each case the robot is operating in the LS2 case where there are close obstacles on both sides of the robot.

The collision and other close brushes with obstacles using ND+ result from the combination of two decisions in the LS2 case handling. To provide a smoother bridge between the HSNR (no close obstacles, but V_{best} is narrow) case and LS2, ND+ always uses θ_{mid} in LS2 regardless of the width of V_{best}. In addition, the two deflection terms for the closest obstacle on the left and right of the robot are averaged instead of being weighted by relative proximity of the obstacles.

When the robot collided with the environment, θ_{srg} is located directly in front of it while θ_{mid} is angled to the

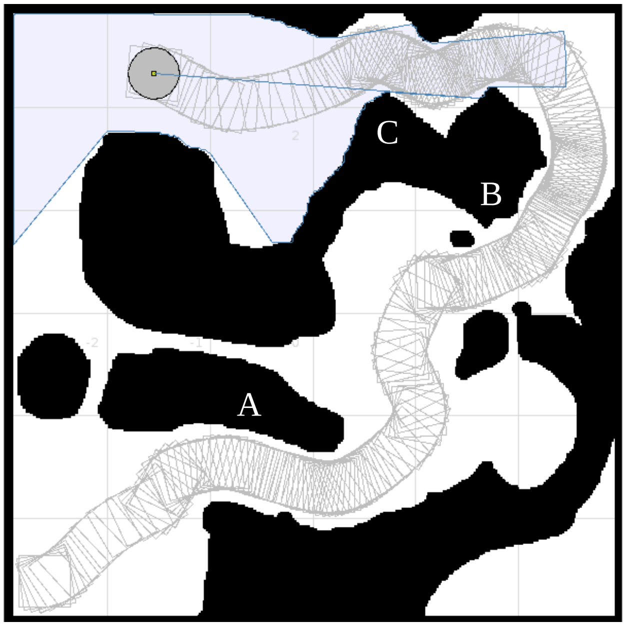

Fig. 3. Simulation results showing path followed by SND method through tight obstacles.

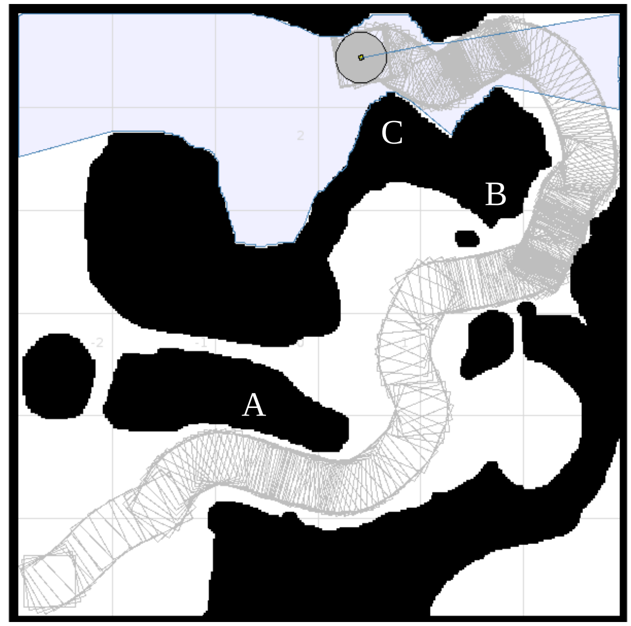

Fig. 4. Simulation results showing path followed by ND+ method using the same conditions as Fig. 3. Constraints on the selection of θ_{d} and the weighting of deflection angles in the LS2 case of ND+ cause the robot to collide with the environment and get stuck.

right. Though the robot is touching the right obstacle and the deflection angle δ_R points directly away from the obstacle, its influence is divided by two when averaged with the δ_L. The combined pull to the right of θ_{mid} and $\delta_L/2$ is equal to $\delta_R/2$ and the robot does not avoid the collision.

D. Comparison of Paths Generated

To demonstrate the increased smoothness of the paths generated by SND, we recorded θ_{traj} over the course of these simulations. In Fig. 5, θ_{traj} is shown for (a) SND and (b) ND+ and is plotted against the distance traveled by the robot so that points on the graphs roughly correspond. While traveling through the open starting area the two methods are fairly similar but differences are clear once they enter the tight corridor labeled A. The sharp changes in θ_{traj} for ND+ in this corridor are the result of considering only the closest obstacle point on the left and right of the robot. In tight scenarios with many obstacles points, the direction towards the closest point will change frequently as the robot moves. These frequent changes cause the many sharp turns in Fig 5(b) near A. By using a weighted sum over all the obstacle points, SND avoids these sharp changes in θ_{traj} while still emphasizing the closest points. Similar improvements in smoothness near obstacles can be seen at B and C.

While leaving the A corridor, ND+ suffers from several large spikes in θ_{traj}. Most of these spikes are the result of gaps merging or disappearing, smaller sharp changes can also be seen in the SND plot. As mentioned in Subsection III-B, both ND+ and SND use the same mechanisms for determining which valley is the best to follow, the differences are in picking θ_{srg} or θ_{mid} and then adjusting due to obstacles. The spikes are larger for ND+ because under the LS2 condition it always chooses θ_{mid} regardless of the width of V_{best}. When gaps merge or split and the valley width changes suddenly, θ_{mid} jumps as well. By selecting whichever of θ_{srg} or θ_{mid} is closer to θ_{rg}, SND reduces these effects. ND+ also shows sharp changes in behavior when transitioning between LS2 and LS1 where it switches from following θ_{mid} to θ_{srg}.

V. EXPERIMENTAL RESULTS

For these experiments we used the Erratic mobile robot platform from Videre Design with an on-board computer and a Hokuyo URG-04LX laser rangefinder. The vehicle platform is roughly square $(40cm \times 37cm)$ with two differential drive wheels and a single rear caster. The laser scans 683 points over $240°$ at $10Hz$. The on-board computer runs the same SND code used for the simulations in Section IV through Player/Stage. The $1.8GhzCore2Duo$ processor runs the reactive SND method in less than $10msec$ and we then wait for the $10Hz$ updates from the laser.

Fig. 6 shows a picture of the robot navigating an obstacle course. A video of the SND experiment is also included in the submission of this paper. As shown in Fig. 7, the SND method in (a) produces smoother changes in heading while avoiding obstacles than the ND+ method shown in (b). Increases in sharp transitions when compared to the simulations are expected since the rangefinder has only a $240°$ field of view. By summing δ_i over all obstacle points, SND reduces the sharpness of these field of view effects, in addition to the other improvements mentioned in Subsection IV-D.

VI. CONCLUSIONS AND FUTURE WORKS

We have presented the Smooth Nearness-Diagram (SND) local navigation method for reactive obstacle avoidance.

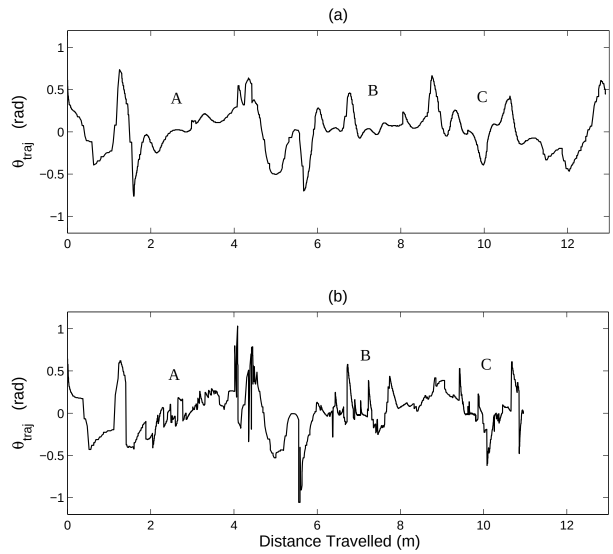

Fig. 5. Comparison of the angular heading θ_{traj} output by (a) SND (b) ND+ along the paths shown in Fig. 3 and 4 respectively.

Fig. 6. Erratic mobile robot navigating an obstacle course using SND.

SND improves the smoothness of paths generated by Nearness-Diagram methods by creating a single motion law for all scenarios and by taking all nearby obstacles into account instead of just the closest two. Comparisons between SND and ND+ in numerical simulations and in experiments with ground robots demonstrated this improvement.

Future work will focus on two research objectives. First, expanding upon our observations and experimental evidence, it is of interest to prove that the SND control law depends continuously on the position of the robot and the environment. Second, more analysis is needed to determine the circumstances under which SND is guaranteed to find safe paths through an environment.

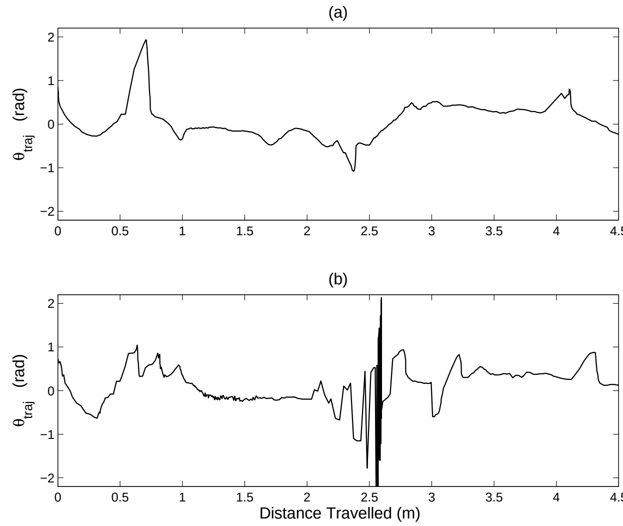

Fig. 7. Theta trajectory plots for the experiment using (a) SND (b) ND+.

References

[1] B. A. Maxwell, W. Smart, A. Jacoff, J. Casper, B. Weiss, J. Scholtz, H. Yanco, M. Micire, A. Stroupe, D. Stormont, and T. Lauwers, "2003 AAAI robot competition and exhibition," *AI Magazine*, vol. 25, no. 2, pp. 68–80, 2004.

[2] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Robotic Systems*, vol. 23, no. 9, pp. 661–692, 2006.

[3] 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.

[4] B. Tovar, L. Guilamo, and S. M. LaValle, "Gap navigation trees: Minimal representation for visibility-based tasks," in *Algorithmic Foundations of Robotics VI* (B. Siciliano, O. Khatib, and F. Groen, eds.), vol. 17 of *Springer Tracts in Advanced Robotics*, pp. 425–440, New York: Springer Verlag, 2005.

[5] B. Tovar, R. Murrieta-Cid, and S. M. LaValle, "Distance-optimal navigation in an unknown environment without sensing distances," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 506–518, 2007.

[6] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, 2004.

[7] L. Montesano, J. Minguez, and L. Montano, "Lessons learned in integration for sensor-based robot navigation systems," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, pp. 85–91, 2006.

[8] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, pp. 354–363, 6 2005.

[9] I. Ulrich and J. Borenstein, "VFH*: Local obstacle avoidance with look-ahead verification," in *IEEE Int. Conf. on Robotics and Automation*, (San Francisco, CA), pp. 2505–2511, Apr. 2000.

[10] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[11] A. Ganguli, J. Cortés, and F. Bullo, "Maximizing visibility in nonconvex polygons: Nonsmooth analysis and gradient algorithm design," *SIAM Journal on Control and Optimization*, vol. 45, no. 5, pp. 1657–1679, 2006.

[12] B. Gerkey and contributors, "Player/Stage/Gazebo Project." `http://www.sourceforge.net/playerstage`, Sept. 2007. version 2.03.