# Self-Tuning Nearness Diagram Navigation

Chung-Che Yu, Wei-Chi Chen, Chieh-Chih Wang and Jwu-Sheng Hu

***Abstract*— The nearness diagram (ND) navigation method is a reactive navigation method used for obstacle avoidance in which five different robot-environment states are defined and five corresponding actions are designed carefully. In the ND+ navigation method, one more robot-environment state is added and all action equations are reformed to both achieve smoother robot motions and reduce manual parameter tuning. In this paper, we argue that the original five states of the ND navigation method is sufficient with the proposed self-tuning method. Two states' action rules of the ND navigation method are modified. The wiggling motion in long corridors is significantly reduced, and a parameter is self-tuned so that manual parameter tuning is avoided. The experimental results using a real robot with a laser scanner demonstrated that the proposed self-tuning ND navigation (stND) approach is simple yet effective.**

## I. Introduction

Safely navigating through previously known or unknown environments is one of the most fundamental requirements for mobile robots. A number of the collision free navigation approaches have been developed such as the potential field method [1], vector field histogram [2], dynamic window approach [3], and elastic band [4]. Recently, the nearness diagram (ND) navigation method proposed by Minguez and Montano [5][6] has showed satisfactory results in terms of obstacle avoidance in crowded and troublesome environments. The ND navigation method is a reactive navigation approach. First, a set of complete and exclusive situations or robot-environment states are defined manually. Second, these predefined situations are represented in a decision tree by analyzing the relationships. The robot's state is then categorized into one of these predefined situations using a decision-tree method [6][7] according to laser scanner data during operation. For each situation, the robot will follow the corresponding strategy to calculate its velocity and angular velocity for accomplishing the task.

As the ND navigation method is reactive, the robot actions can be not smooth often. The wiggling motions are often observed even the robot is controlled using the ND navigation method with a high-level path planner. It is possible to fine-tune some parameters of the ND navigation method to improve the smoothness of the robot motion [6]. In the newer version of ND navigation method (ND+), one more situation is added and all action equations are reformed to both achieve smoother robot motions and reduce manual parameter tuning [7]. In this paper, we argue that the original five states of the ND navigation method is sufficient with the proposed self-tuning method. Two states' action rules of the ND navigation method are modified. The wiggling motion in long corridors is significantly reduced, and a parameter is self-tuned so that manual parameter tuning is avoided. It could be argued that the wiggling motion could be improved with high level path planning algorithms as dense waypoints can be generated to guide the robot to act smoother. But even with a good sub-goal generation approach, an accurate robot localization system is critical, and could be problematic and computational expensive in practice. The proposed approach does not need any global localization systems and high-level path planners to perform smoother motion. The experiments were conducted using a real robot with a laser scanner in a long corridor as depicted in Figure 1. The results demonstrated that the proposed self-tuning ND (stND) navigation approach is simple yet effective.

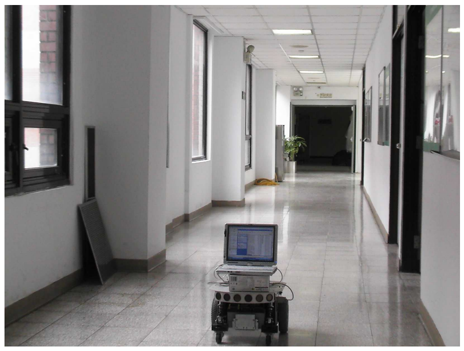


Fig. 1. The NTU-PAL5 robot with a SICK LMS291 laser scanner was used to verify the proposed approach in a long corridor.

The rest of this paper is organized as follows. The ND and ND+ navigation methods are briefly introduced in Section II. In Section III, the proposed approach is described in detail. The experimental results are shown in Section IV. The pros and cons of the proposed approaches and the existing approaches are discussed in Section V. Finally, our conclusions and future works are presented in Section VI.

## II. The ND and ND+ Navigation Methods

In the section, the ND and ND+ navigation methods are briefly reviewed and critical issues are discussed.

Chung-Che Yu and Wei-Chi Chen are with the Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei, Taiwan `fish60@robotics.csie.ntu.edu.tw`

Chieh-Chih Wang is with the Department of Computer Science and Information Engineering, and the Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei, Taiwan `bobwang@ntu.edu.tw`

Jwu-Sheng Hu is with the Department of Electrical and Control Engineering, National Chiao-Tung University, and with the Mechanical and Systems Research Laboratories, Industrial Technology Research Institute, Hsin Chu, Taiwan `jshu@cn.nctu.edu.tw`

## A. ND Navigation

Recall that the ND method first defines a set of complete and exclusive situations and then uses these situations to simplify the difficulty of robot navigation. The key of the ND navigation method is to use a *divide and conquer* strategy by setting criterions to identify the current situation of the robot. The first criterion is the safety criterion. If there is any obstacle within the predefined security zone, the robot would be in the *Low Safety* state otherwise in the *High Safety* state. Based on the obstacle distribution within the security zone, the Low Safety state is further classified to in two classes: the Low Safety 1 (LS1) state if the obstacles in the security zone are only on one side of the gap (closest to goal) of the free walking area, and the Low Safety 2 (LS2) state in which the both sides are occupied with obstacles. The High Safety state are further classified into three states. If the goal is within the free walking area, it is in the High Safety Goal in Region (HSGR) state. Otherwise, checking the width of the free walking area, the robot would fall into the High Safety Wide Region (HSWR) or the High Safety Narrow Region (HSNR).

In the ND navigation method, there are several important parameters which should be chosen carefully. Some parameters are related to the robot information such as the shape, the maximum velocity, and the maximum angular velocity of the robot. Others are the implementation parameters such as the safety distance, the bound of the angular width of a narrow region, and the parameter for ensuring a smooth behavior in the transitions among the situations. The robot could suddenly change its motion just because of bad parameters.

## B. ND+ Navigation

Two of the most important differences between ND and ND+ are the new state for Low Safety and the reduction of manual parameter tuning. The added situation is the Low Safety Goal in Region (LSGR) in which there are obstacles within the security zone, and the goal location is within the free walking area. For more detailed discussion, see the Section V.

Figure 2 illustrates the fundamental differences between the ND and ND+ navigation approaches, and also shows the focuses of the proposed self-tuning ND (stND) navigation which are addressed in detail in the next section.

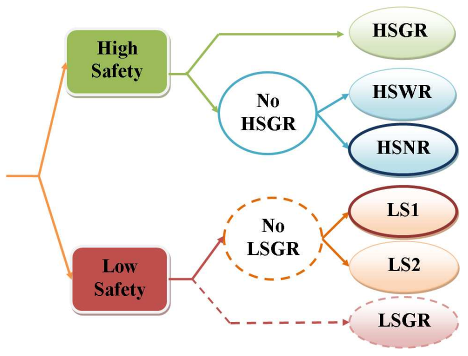


Fig. 2. The first five states (solid line) are defined by the original ND navigation method, whereas the ND+ navigation method define a new criterion and the sixth state (dash line). Our approach uses only the first five state and focus on the modification of actions for High Safety Narrow Region (HSNR) and Low Safety 1 (LS1) states (bold line).

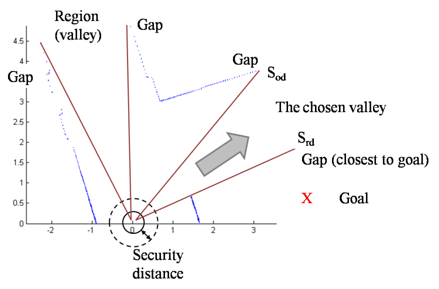


Fig. 3. The definitions of gaps, valleys, the free walking area, $S_{rd}$, and $S_{od}$. The blue points are raw laser range data. The black circle indicates the robot position. The arrow indicates the chosen valley or the free walking area.

## III. Self-Tuning ND Navigation

Intuitively, the action rules of HSGR, HSWR, and HSNR should deal with the safer situations, and the polices of LS1 and LS2 could handle low safety situations in crowded or troublesome environments. However, the ND navigation approach may have poor performance in terms of motion smoothness. For instance, the robot motion can often wiggle in long corridors as the robot states often transit between the states of High Safety and Low Safety.

In this section, the proposed approach is described based on the original ND navigation method. We intended to use a method, which is as simple as possible, to solve the wiggling motion problem for driving the robot straight in long corridors. First, the action rule of one state in the ND navigation method is modified. Second, an online tuning strategy for the parameter $p$ is proposed. These two modifications make the path of the robot looks smoother and more straight than the original ND navigation approach.

## A. Important Terms from ND Navigation

This paper follows the notations defined in [6]. The related variables are introduced for understanding the proposed approach. A region is called a *gap* in which there are two contiguous range measurements are either separated by more than the robot diameter $2R$ or one of the measurements returns no obstacle in range. See [6] for the detail information on calculating the locations of gaps. Each pair of consecutive gaps would define a *region*. The navigable region of the robot is called a *valley*. For the valley which defined by the gap closed to the goal, the direction information of that gap is $S_{rd}$. The direction information of the other gap of that chosen valley, or the free walking area, is $S_{od}$. These variable definitions are illustrated in Figure 3.

## B. Action Design of High Safety Narrow Region (HSNR)

The definition of the High Safety Narrow Region (HSNR) is that the robot is in a free walking area whose angular width is smaller than a given angle. The action design for HSNR

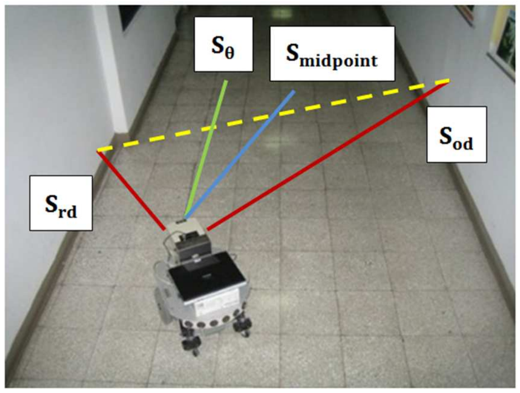


Fig. 4. The red lines show $S_{rd}$ and $S_{od}$ of the chosen valley of free walking area. The green line presents the bisector approach in ND which is the bisector of $S_{rd}$ and $S_{od}$. The blue line presents the midpoint approach in stND which passes through the midpoint of yellow line. It could be observed that the blue line passes the center zone of the corridor better than the green line.

is to direct the robot through the center zone of the free walking area. In the original ND navigation implementation, the direction of the robot is computed as the bisector of the direction of the discontinuities of the selected valley. However, the robot motion using this rule may not be straight enough in long corridors. It is observed that the robot would not try to pass through the center of the corridor if the robot's heading angle is not parallel enough with the corridor.

The high level path planner could generate a sequence of waypoints to guide the robot moving through the center of the corridor. But we intended to use low level control and simple rules to get a robust result. It is believed that the more robust the low level control is, the better performance could be achieved when high level planning involves in.

In ND, the direction information of the chosen valley is used to get the resulting action angle. In detail, the $S_{rd}$ is the sector corresponding to the rising discontinuity (closest to the sector that contains the goal location) and the $S_{od}$ is the sector corresponding to the other discontinuity of the selected valley. The resulting action of HSNR in ND is computed by:

$$S_\theta = \frac{S_{rd} + S_{od}}{2} \quad (1)$$

The final heading sector $S_\theta$ is further used to compute the translation velocity. In stND, the proposed action design for HSNR is to compute the direction using not only the direction but also the range information of $S_{rd}$ and $S_{od}$. In detail, we calculate the midpoint of range data of $S_{rd}$ and $S_{od}$, and then the motion direction is set by robot's current position and the midpoint. The action rules of HSNR in ND and stND are illustrated in Figure 4.

## C. Self-Tuning Method of Parameter $p$ in LS1

With the modification of the action design in HSNR, a more smooth and straight motion could be achieved. But there are some parameters which should be carefully chosen such as the parameters related to the robot itself. However, in long corridors, the reactive method is highly sensitive to the parameter $p$ of LS1 which ensures a smooth behavior in the transitions among the robot states.

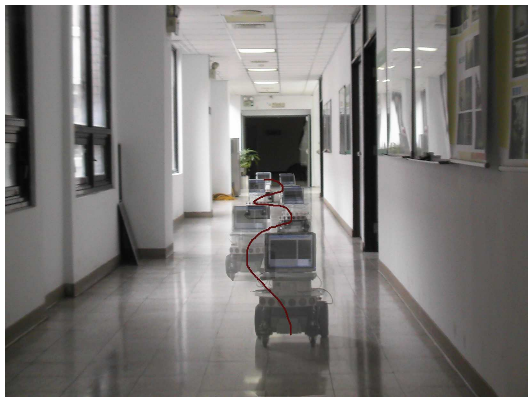


Fig. 5. The robot wiggles at a high speed through the long corridor with a big $p$ in LS1.

Recall that the robot is in the LS1 state when the obstacles in the security zone are only on one side of the gap (closest to the goal) of the free walking area. The objectives of action design of LS1 are to move the robot away from the closest obstacle, and to move toward the gap of the free walking area. The parameter $p$ of LS1 effects the turning rate of the robot for moving away from the closest obstacle. The equation to compute the heading angle of the robot being in the LS1 state of ND is described below.

$$S_p = |S_{rd} - S_{ml}| * p + \frac{S_{max}}{2} \quad (2)$$

$$S_\theta = S_{rd} + sign(S_{rd} - S_{ml}) * S_p \quad (3)$$

where $S_{max}$ is a given static value and $S_{ml}$ is the closest obstacle direction. The term $sign(S_{rd} - S_{ml})$ indicates the right turn or left turn. The value of $S_p$ controls the turning level of the robot. $S_\theta$ is composed by $S_{rd}$ which contains the goal direction information and the obstacle avoidance turning angle. The magnitude of the parameter $p$ would effect the turning angle. The proposed approach here eliminates the term $S_{max}/2$ but uses a self tuning method to adjust the $p$ value. The new equation of $S_p$ becomes:

$$S_p = |S_{rd} - S_{ml}| * p \quad (4)$$

The experimental results show that the robot will make a big turn in the LS1 state if $p$ is too big. This behavior occurs often in long corridors.

This wiggling motion is because the values of maximum velocity and angular velocity are high, and the value of $p$ is also too high. In other words, the robot is too sensitive to get rid of the closest obstacle. Even though avoiding obstacles is critical important, the robot could act strangely if the robot motion is too sensitive to the environment. Figure 5 shows an experimental result in which the robot is wiggling at a high speed and angular velocity with a relative high $p$. On the other hand, the robot may lose the ability to escape from LS1 if the value of $p$ is set too low. Therefore, a suitable value of $p$ based on the environment should be found online given that the other parameters such as maximum velocity and angular velocity are set. The idea is simple yet effective. A low value of $p$ is chosen for more straight motion and the value of $p$ can be increased if necessary. The self-tuning pseudo code is described in Algorithm 1.

**Algorithm 1** The Online Tuning Algorithm of $p$

```
Require: $(State_{pre}, State_{now})$ here, $State_{pre}$/$State_{now}$
    means the previous/current state of robot which is one
    of the states defined by the ND method
1: Initialize $p$ with the minimum value if it is the first call
2: if $State_{pre}$ is LS1 then
3:     if $State_{now}$ is LS1 then
4:         Call $IncreaseP()$
5:     else
6:         Call $DecreaseP()$
7:     end if
8: end if
9: $State_{pre} \leftarrow State_{now}$
```

The function $IncreaseP()$ and $DecreaseP()$ in Algorithm 1 online tune the value of $p$ between the predetermined upper bound and lower bound of $p$. If the calculated value would be lower than the lower bound or exceed the upper bound, these functions would set the $p$ value to the boundary value. Different implementations would effect the growing and shrinking rate of $p$. Below our approaches are described in detail.

*1) Increasing Strategy:* One of the easiest ways to construct the function $IncreaseP()$ is to add a static value to $p$ while this function is called. The static value could be chosen according to experiments and observations. However, in the other way, the increasing step is directly related to the robot's ability of obstacle avoidance. It could be decided that how soon the robot should use the highest value to escape from the LS1 state. Therefore, the function $IncreaseP()$ can be described as:

$$p_n = p_1 + (n-1)d \tag{5}$$

This approach presents the $p$ value as one of the arithmetic sequence in which the initial term $p_1$ is the minimum value of $p$. The symbol $d$ stands for the positive common difference here.

It should be noted that the arithmetic sequence would not perform well if the robot is in LS1 for a long period of time. That is, the growing rate of $p$ could not quickly escape from the LS1 state into the high safety states. Thus, the longer the robot is in LS1 state, the higher growing rate of $p$ should be. The geometric sequence approach can be applied.

$$p_n = p_1 * q^{(n-1)} \tag{6}$$

where the variable $q$ stands for the common ratio which is greater than 1. The growing rate of $p$ would be slower using common ratio at the beginning but be faster after some certain iterations.

*2) Decreasing Strategy:* After the robot left the LS1 state, the $p$ value should be decreased to ensure that a relative low turning angle and velocity are applied when the robot enters the LS1 state again. One of the simplest ways to implement the function $DecreaseP()$ is to reset $p$ to the initial value, the minimum value of $p$. Recall that we would like to avoid the wiggling behavior in long corridors. Resetting $p$ could guarantee that the minimum turning angle is used to guide the robot to the center of the corridor smoothly.

It is also feasible to design the decreasing strategy following the same idea of the increasing strategy but with the reverse action. The $p$ value can be decreased by one common difference using the arithmetic sequence approach or by dividing the $p$ value with a common ratio in the geometric sequence approach. Compared to the reset approach, these methods would let the robot have a higher $p$ value during the operation. With the higher value of $p$, the robot could have better performance in troublesome environments.

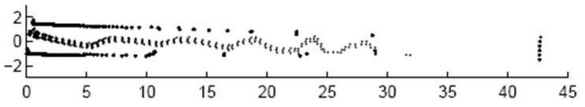
## IV. Experimental Results

A differential-drive robot with a SICK LMS291 laser scanner was used to verify the proposed stND navigation approach. The width of the robot is 44 centimeters and the length is 52 centimeters. The testing environment is a long corridor (2.5 meters x 43.0 meters) as depicted in Figure 1). One extra SICK LMS291 laser scanner was used and located on the left side of the corridor for recording the robot trajectories and providing ground truth. The ND navigation method with the proposed midpoint approach for HSGR, the stND navigation method with the proposed midpoint approach for HSGR and with the arithmetic sequence approach using common differences 0.10 and 0.05 for LS1 are implemented and compared.

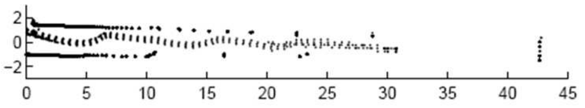
Figure 6 (a) shows the robot trajectories using the ND navigation method with the proposed midpoint approach for HSNR only in which the robot moved from the left end of the corridor to right. The states of the robot are shown in Figure 7 (a) in which the number of HSNR state is very few so it could be seen as the original ND navigation method. Figure 6 (b)(c) and Figure 7 (b)(c) show the experimental results in which the robot moved from the right end of the corridor to left with different settings. As a laser scanner and other equipments were located on the left side of the corridor which create two narrow passageways, the collected scan data are quite different between the right-to-left and left-to-right test runs. Thus we could use the figure 6 (a) as original ND result and compare it with the proposed method.

Figures 6 demonstrate that the proposed stND navigation approaches perform less wiggling motion than the ND navigation method. The robot has fewer big turns with the use of the proposed methods. Further comparing (b) and (c) of Figures 6, the robot is nimbler in (b) but gets a slightly wiggly trajectory than (c). This result shows that the stND navigation method could ease the wiggling level in long corridors.

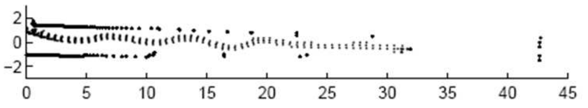
It can be observed that the robot would stay less away from the center zone of the corridor with the use of the proposed



(a) ND with the proposed midpoint approach for HSNR

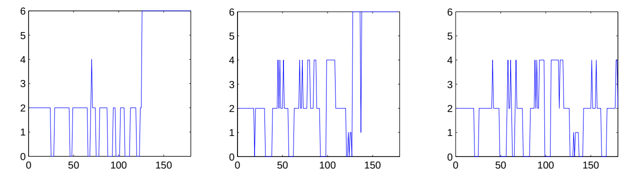


(b) stND with a common difference 0.10 for $p$ in LS1



(c) stND with a common difference 0.05 for $p$ in LS1

Fig. 6. The robot trajectories using different approaches in which the robot moved from the left end of the corridor to right in (a) and from the right end of the corridor to left in (b) and (c).



(a) ND with the proposed midpoint approach for HSNR (b) stND with a common difference 0.10 for $p$ in LS1 (c) stND with a common difference 0.05 for $p$ in LS1

Fig. 7. The robot states corresponding to the results in Figure 6. The state number 0 to 6 stands for: LS1, LS2, HSGR, HSWR, HSNR, no path, and too close. The last two states are added to include the situations that the robot can not find a path and there is an emergency to stop the robot. The x-axis indicates the iterations of the robot which costs approximately 0.25 second for one iteration.

modification of HSNR as depicted in Figure 7. Note that the proposed approach would ease the burden of high level path planning without many waypoints generation.

For ensuring the proposed method still could work in crowded environment, we also test our algorithm in the environment other than long corridor. Figure 8 demonstrates that the proposed stND navigation approaches perform nice result in crowded environment. Figure 9 is the corresponding state of the result.

## V. Discussions

TABLE I
The comparison between ND, ND+, stND

| | ND | ND+ | stND |
|---|---|---|---|
| Number of states | 5 | 6 | 5 |
| Use parameter p | Yes (static) | No | Yes (self-tuned) |

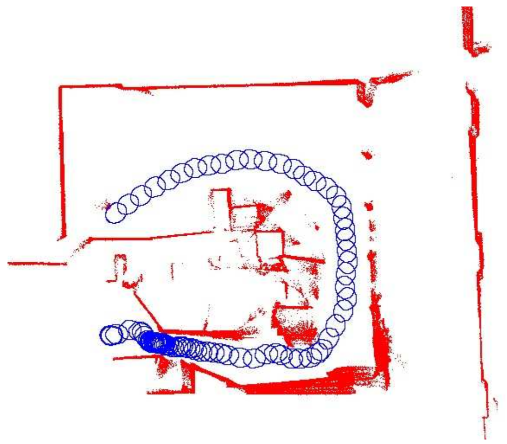


Fig. 8. The robot trajectory in crowded environment with a very narrow passage.

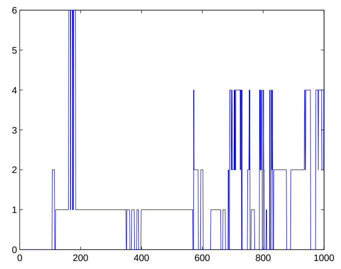


Fig. 9. The robot states corresponding to the results in Figure 8. The state number 0 to 6 stands for: LS1, LS2, HSGR, HSWR, HSNR, no path, and too close.

TABLE II
The comparison of actions of ND, ND+, and stND. From top to bottom: HSGR, HSWR, HSNR. We transfer sector information into angle for robot's heading: $\theta = bisector(S)$

| HSGR | ND | ND+ | stND |
|---|---|---|---|
| $\theta_{sol}$ | $\theta_{goal}$ | $\theta_{goal}$ | $\theta_{goal}$ |

| HSWR | ND | ND+ | stND |
|---|---|---|---|
| $\theta_{dir1}$ | $\theta_{rd}$ | $\theta_{rd}$ | $\theta_{rd}$ |
| $\theta_{dir2}$ | $\theta_{rd}$ | $\theta_{rd}$ | $\theta_{rd}$ |
| $\theta_{pre}$ | $\frac{\theta_{max}}{2}$ | $\arcsin(\frac{R+D_s}{D_{rd}})$ | 0 |
| $\theta_{dev1}$ | 0 | 0 | 0 |
| $\theta_{dev2}$ | 0 | 0 | 0 |
| $\theta_{sol}$ | $\frac{\theta_{dir1}+\theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \theta_{rd} \pm \frac{\theta_{max}}{2}$ | $\frac{\theta_{dir1}+\theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \theta_{rd} \pm \arcsin(\frac{R+D_s}{D_{rd}})$ | $\frac{\theta_{dir1}+\theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \theta_{rd}$ |

| HSNR | ND | ND+ | stND |
|---|---|---|---|
| $\theta_{dir1}$ | $\theta_{rd}$ | $\theta_{rd}$ | $\theta_{rd}$ |
| $\theta_{dir2}$ | $\theta_{od}$ | $\theta_{od}$ | $\theta_{od}$ |
| $\theta_{dev1}$ | 0 | 0 | 0 |
| $\theta_{dev2}$ | 0 | 0 | 0 |
| $\theta_{sol}$ | $\frac{\theta_{dir1}+\theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \frac{\theta_{rd}+\theta_{od}}{2}$ | $\frac{\theta_{dir1}+\theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \frac{\theta_{rd}+\theta_{od}}{2}$ | $\theta_{midpoint(S_{rd},S_{od})} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \theta_{midpoint(S_{rd},S_{od})}$ |

In this section, the pros and cons of the proposed approaches and the existing approaches are compared and discussed. We will use Table I, II, and III to fully compare the proposed stND navigation method with ND and ND+.

The Table I shows the basic comparison between the three approaches. The proposed method use only five states which is the same as ND method whereas the ND+ has the sixth state (LSGR). ND method has parameter $p$ for LS1 state but the user has to experimentally tune the value to get a smooth result. In ND+, it adjusts the deviation of object avoidance using the distance as weighting to eliminate the parameter $p$. In stND navigation method, we keep the parameter $p$ but use a self-tuning method to achieve the smoothness. The detailed mathematical equations and the value of each terms are listed in Table II and III. We will then compare each methods with different states.

First, if the robot is in the HSGR state, the $\theta_{sol}$ would be set to $\theta_{goal}$, which would be the final setting of the robot heading. This is the same for all methods. Otherwise, except the case of stND navigation method in the HSNR state, the $\theta_{sol}$ would be calculated as:

$$\theta_{sol} = \frac{\theta_{dir1} + \theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1} + \theta_{dev2}}{2}) \qquad (7)$$

For the stND navigation method in the HSNR state, the $\theta_{sol}$ becomes:

$$\theta_{sol} = \theta_{midpoint(S_{rd},S_{od})} \pm (\theta_{pre} + \frac{\theta_{dev1} + \theta_{dev2}}{2}) \qquad (8)$$

The $\theta_{sol}$ in Eq. (7) and Eq. (8) contains three parts. First part is the heading direction for approaching the goal, which is $\frac{\theta_{dir1}+\theta_{dir2}}{2}$ for all other cases except in the HSNR state for the stND navigation method whereas the stND navigation method uses $\theta_{midpoint(S_{rd},S_{od})}$ when robot is in the HSNR state, which is one of the contributions of this paper. Then the second and third part are the deviation of the robot heading, which are used for keeping robot away from obstacles. $\theta_{pre}$ is used to keep a sufficient distance away from nearby obstacles to maintain robot in high safety state after the robot moving along the final direction with some certain distance. The last one, $\frac{\theta_{dev1}+\theta_{dev2}}{2}$, is used to avoid obstacles in order to escape low safety state or guide robot not to hit the obstacles nearby during the operation.

The $\theta_{sol}$ would be the result most of the times, but sometimes there may be some adjustments based on multiple heading choices. So, we may use $\theta_{sol'}$ for the final heading choice if needed. We would use $\theta_{closerTo\theta_{rd}}(.)$ which could take multiple inputs and output the result which is closer to the specified $\theta_{rd}$. We may then adjust it to get the $\theta_{sol'}$.

We then carefully analyze the rest of the states: HSWR, HSNR, LS1, LS2, and LSGR using Table II and III. First, we check the HSWR state. Since the robot is in high safety state, the basic heading information is contained only in $\theta_{rd}$, and the $\theta_{dir1}$ and $\theta_{dir2}$ for all methods are the same. The third part, which consists $\theta_{dev1}$ and $\theta_{dev2}$, for all methods is set to zero since the robot is not in emergency. The main difference between them in the HSWR state is $\theta_{pre}$. In original ND method, it uses a pre-defined value $\frac{\theta_{max}}{2}$ to keep robot in high safety state. Whereas in ND+ the radius of the robot $R$, safety distance $D_s$, and the distance to the rising discontinuity $D_{rd}$ are combined to achieve the same goal. The radian $\arcsin(\frac{R+D_s}{D_{rd}})$ ensures the robot center to have the distance $R+D_s$, which is the decision boundary of low safety or high safety, away from obstacle which creates the $\theta_{rd}$ at first after robot moving along the direction $\theta_{sol}$ with distance $D_{rd} * \cos(\arcsin(\frac{R+D_s}{D_{rd}}))$. In stND method, we just set the value as zero because even without the additional deviation and the robot may enter the LS1 state, the action of LS1 could keep the robot away from obstacles and maintain the smoothness of the trajectory.

Second, we check the HSNR state. The values of $\theta_{dir1}$, $\theta_{dir2}$, $\theta_{pre}$, $\theta_{dev1}$, and $\theta_{dir2}$ for ND, ND+, and stND navigation methods are all the same. Because the region is narrow, the goal now is to guide the robot through the center zone of the chosen region and we set the value of $\theta_{pre}$ to zero. $\theta_{dev1}$ and $\theta_{dev2}$ for all methods are also set to zero since the robot is not in emergency. The ND and ND+ navigation methods use the bisector result whereas the proposed method uses the midpoint approach mentioned in Section III-B. The discussion of the modification of the HSNR state has been showed in Section III-B and this modification would not cause robot fail since the robot is in a high safety state.

Which we want to address here is that both the ND+ navigation and the smooth nearness diagram (SND) navigation method [9] do not use the midpoint of range data of the left and right valleys to guide the robot. In addition, the proposed method applies simple yet effective strategies tries to improve ND. Instead of using the information of the closest obstacle around the robot, the SND navigation method uses the information of all obstacles for improving motion smoothness. We have shown that the proposed method could still get nice results using just the information of the closest obstacle.

Then we check the low safety states. The action for the LS1 state is to avoid the closest obstacle while still approach the goal. The $\theta_{dir1}$ and $\theta_{dir2}$ for ND, ND+, and stND navigation methods are the same thus they get the same direction from goal information. However, the $\theta_{pre}$, $\theta_{dev1}$, and $\theta_{dev2}$ are different. As the same in HSWR, the ND navigation method uses $\frac{\theta_{max}}{2}$ for $\theta_{pre}$ which is a pre-defined value. As the same reason from HSWR, the ND+ navigation method uses $\arcsin(\frac{R+D_s}{D_{rd}})$ for $\theta_{pre}$. Then they further adjust the robot heading using $\theta_{dev1}$ and $\theta_{dev2}$. The ND uses the information of the closest obstacle and an experimentally tuned parameter $p$ for obstacle avoidance and the ND+ considers the deviation with the distance and direction of the closest obstacle and the safety distance to eliminate the parameter $p$. However, the proposed method set the $\theta_{pre}$ to zero. We then use a online self-tuning $p$ to guide the robot to escape from low safety state to high safety state and maintain the smoothness between LS1 and high safety states as mentioned in Section III-C. Here, the $\theta_{dev1}$ and $\theta_{dev2}$ are the same value since the obstacle in security zone occupied only one side of the robot. The experiments show

TABLE III

THE COMPARISON OF ACTIONS OF ND, ND+, AND STND. FROM TOP TO BOTTOM: LS1, LS2, LSGR. THE $D_{obs}$ IS THE DISTANCE OF THE CLOSEST OBSTACLE TO THE BOUNDARY OF THE ROBOT AND THE $\theta_{obs}$ IS THE ANGLE RESPECT TO THE CENTER OF THE ROBOT

| LS1 | ND | ND+ | stND |
|---|---|---|---|
| $\theta_{dir1}$ | $\theta_{rd}$ | $\theta_{rd}$ | $\theta_{rd}$ |
| $\theta_{dir2}$ | $\theta_{rd}$ | $\theta_{rd}$ | $\theta_{rd}$ |
| $\theta_{dev1}$ | $\lvert\theta_{rd} - \theta_{ml}\rvert * p_{static}$ | $\frac{D_s - D_{obs}}{D_s} \cdot \lvert(\pi + \theta_{obs}) - \lvert\theta_{dir1} - \theta_{pre}\rvert\rvert$ | $\lvert\theta_{rd} - \theta_{ml}\rvert * p_{selftuned}$ |
| $\theta_{dev2}$ | $\lvert\theta_{rd} - \theta_{ml}\rvert * p_{static}$ | $\frac{D_s - D_{obs}}{D_s} \cdot \lvert(\pi + \theta_{obs}) - \lvert\theta_{dir2} - \theta_{pre}\rvert\rvert$ | $\lvert\theta_{rd} - \theta_{ml}\rvert * p_{selftuned}$ |
| $\theta_{sol}$ | $\frac{\theta_{dir1}+\theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \theta_{rd} \pm (\frac{\theta_{max}}{2} + \lvert\theta_{rd} - \theta_{ml}\rvert * p_{static})$ | $\frac{\theta_{dir1}+\theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \theta_{rd} \pm (\arcsin(\frac{R+D_s}{D_{rd}}) + \frac{D_s - D_{obs}}{D_s} \cdot \lvert(\pi + \theta_{obs}) - \lvert\theta_{rd} - \arcsin(\frac{R+D_s}{D_{rd}})\rvert\rvert)$ | $\frac{\theta_{dir1}+\theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \theta_{rd} \pm \lvert\theta_{rd} - \theta_{ml}\rvert * p_{selftuned}$ |

| LS2 | ND | ND+ | stND |
|---|---|---|---|
| $\theta_{dir1}$ | $\theta_{ml}$ | $\theta_{rd}$ | $\theta_{ml}$ |
| $\theta_{dir2}$ | $\theta_{mr}$ | $\theta_{od}$ | $\theta_{mr}$ |
| $\theta_{dev1}$ | 0 | $\frac{D_s - D_{obs1}}{D_s} \cdot \lvert(\pi + \theta_{obs1}) - \lvert\theta_{dir1} - \arcsin(\frac{R+D_s}{D_{rd}})\rvert\rvert$ | 0 |
| $\theta_{dev2}$ | 0 | $\frac{D_s - D_{obs2}}{D_s} \cdot \lvert(\pi + \theta_{obs2}) - \lvert\theta_{dir2} - \arcsin(\frac{R+D_s}{D_{rd}})\rvert\rvert$ | 0 |
| $\theta_{sol}$ | $\frac{\theta_{dir1}+\theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \frac{\theta_{ml}+\theta_{mr}}{2}$ | $\frac{\theta_{dir1}+\theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \theta_{rd} \pm (\frac{D_s - D_{obs1}}{2*D_s} \cdot \lvert(\pi + \theta_{obs1}) - \lvert\theta_{rd} - \arcsin(\frac{R+D_s}{D_{rd}})\rvert\rvert + \frac{D_s - D_{obs2}}{2*D_s} \cdot \lvert(\pi + \theta_{obs2}) - \lvert\theta_{od} - \arcsin(\frac{R+D_s}{D_{rd}})\rvert\rvert)$ | $\frac{\theta_{dir1}+\theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \frac{\theta_{ml}+\theta_{mr}}{2}$ |
| $\theta_{sol'}$ | $\theta_{closerTo\theta_{rd}}(\theta_{sol}, \theta_{sol} + \pi) \pm c$ | $\theta_{sol}$ | $\theta_{closerTo\theta_{rd}}(\theta_{sol}, \theta_{sol} + \pi) \pm c$ |

| LSGR | ND | ND+ | stND |
|---|---|---|---|
| $\theta_{dir1}$ | See LS1/LS2 | $\theta_{goal}$ | See LS1/LS2 |
| $\theta_{dir2}$ | See LS1/LS2 | $\theta_{goal}$ | See LS1/LS2 |
| $\theta_{pre}$ | See LS1/LS2 | 0 | 0 |
| $\theta_{dev1}$ | See LS1/LS2 | $\frac{D_s - D_{obs}}{D_s} \cdot \lvert(\pi + \theta_{obs}) - \lvert\theta_{dir1} - \theta_{pre}\rvert\rvert$ | See LS1/LS2 |
| $\theta_{dev2}$ | See LS1/LS2 | $\frac{D_s - D_{obs}}{D_s} \cdot \lvert(\pi + \theta_{obs}) - \lvert\theta_{dir2} - \theta_{pre}\rvert\rvert$ | See LS1/LS2 |
| $\theta_{sol}$ | See LS1/LS2 | $\frac{\theta_{dir1}+\theta_{dir2}}{2} \pm (\theta_{pre} + \frac{\theta_{dev1}+\theta_{dev2}}{2}) = \theta_{goal} \pm (\frac{D_s - D_{obs}}{D_s} \cdot \lvert(\pi + \theta_{obs}) - \theta_{goal}\rvert)$ | See LS1/LS2 |
| $\theta_{sol'}$ | See LS1/LS2 | $\theta_{sol}$ | See LS1/LS2 |

good performance of this modification.

For the LS2 state, the final direction should guide the robot in the center of the obstacles and move it to the goal location. Our proposed method follows the ND navigation approach thus the performance would be the same as ND navigation in this situation. The main direction is based on the bisector of the closest obstacles on both sides of the robot. This information is obtained from $\theta_{ml}$ and $\theta_{mr}$. It then chooses the closer one to $\theta_{rd}$ from bisector result and its opposite direction since it implicitly contains the information of the goal location. The ND+ navigation approach uses the bisector of $\theta_{rd}$ and $\theta_{od}$ thus it has the benefit that it does not need to check the opposite direction as ND navigation method whereas the result of direction does not consider the true obstacles around. On the other hand, the bisector result in ND navigation method does not guarantee the robot to drive in the center of the obstacles. So, the $\theta_{dev1}$, and $\theta_{dev2}$ in ND+ navigation method and the correct function $c$ in ND and stND navigation methods both result in the desire of centering the robot between the obstacles. The difference between these two approaches is that using bisector of the obstacles cares more about centering issue first whereas using bisector of $\theta_{rd}$ and $\theta_{od}$ cares more about the goal location. Since both approaches then further consider more about the goal location and obstacles nearby to adjust the final results, the both approaches are satisfied with LS2 case.

LSGR is not listed as one of the states in the ND and stND navigation methods. However, for a fully comparison between ND, ND+, and stND navigation methods, we list it in the Table II. Actually, the LSGR state is a low safety state thus it could be classified into LS1 or LS2, based on the distribution of the obstacles within the security zone. The corresponding action for LSGR is to guide the robot to approach the goal while not to hit the obstacles within the security region. In this sense, the LS1 and LS2 states are sufficient for completing the low safety tasks. The main advantage of LSGR is that it sets $\theta_{pre}$ as zero thus in the low safety goal in region with obstacles just in one side of the security zone one can argue that the result heading would be closer to the goal location compared with LS1 state. However, the proposed method also set $\theta_{pre}$ as zero and use a online self-tuning strategy to reasonably adjust the heading for the LS1 state. For the LS2 state, all of the ND, ND+, and stND set $\theta_{pre}$ as zero so the heading would be good enough. The result would be different between LS2 and LSGR if the free walking area is wide and the goal location is far away from the bisector of $\theta_{ml}$ and $\theta_{mr}$ in ND and stND navigation methods or the bisector of $\theta_{rd}$ and $\theta_{od}$ in ND+ navigation method. We argue that since the free walking area is wide, the robot would then be in HSGR after it escapes from low safety state and then adjust its heading based on the corresponding rules. So, combining with the arguments above, the proposed method does not have the sixth state.

To further compare the stND navigation method with the ND+ navigation method, we need to address their differences in terms of the LSGR and the LS1 states. In the LSGR state of the ND+ navigation method, the direction of the robot motion is set to the direction to the goal location plus a deviation that depends on the distance to the closest obstacle. The closer the distance is, the change of the heading is bigger. Therefore the robot trajectory using ND+ may be not smooth when a close nearby obstacle is firstly encountered. As the action rule of the ND+ LS1 state not only considers the distance to the closest obstacle but also adds an angle to prevent the obstacle from entering the security zone, the heading of the robot in the LS1 state of the ND+ navigation method would have a greater change compared to the LSGR state.

The ND+ navigation method may drive the robot move smoother than the original ND navigation method as LSGR would not have a greater robot heading change. In the proposed stND navigation method, there is no LSGR and robot heading changes are reduced significantly in the LS1 state. In addition, online parameter tuning is applied to reduce wiggling motions and to escape from the low safety states simultaneously. The robot would not have a sudden change in a low safety state but in ND+ the robot heading change using ND+ is big when an obstacle is nearby. We have shown that the original five states of the ND navigation method are sufficient given that the $p$ value is online adjusted nicely.

## VI. Conclusions And Future Works

### A. Conclusions

This paper presented the modification of the ND HSNR state and an online parameter tuning strategy for smooth motions in long corridors. The proposed approach would not lose the ability of the original ND navigation method in open space, and keep the same power when the robot is in troublesome environments. The experimental results show that the proposed stND navigation method drives the robot more straight and smoother in such long corridor environments.

### B. Future Works

It has been shown that the whole scan data could be used to achieve smoother motions [9]. This should be of our interest to exploit this idea. It is also our future work to see if the status of the surroundings could be recognized and used for deciding a proper level of reaction. It should be possible to make a more suitable strategy during online operation with higher level scene understanding. In addition, the idea of learning parameters by observing how human operates a robot [8] will be explored to improve the proposed approaches.

## Acknowledgments

This work was also partially supported by grants from Taiwan ITRI; Taiwan NSC (#96-2628-E-002-251-MY3, #97-2218-E-002-017, #98-2623-E-002-017-D); Excellent Research Projects of National Taiwan University (#98R0062-AE00-05); Taiwan DOIT TDPA Program (#95-EC-17-A-04-S1-054); Taiwan CCI, MSI and Intel.

## References

[1] O. Khatib, Real-Time Obstacle Avoidance for Manipulators and Mobile Robots, *The International Journal of Robotics Research*, 5(1), 1986

[2] J. Borenstein and Y. Koren, The Vector Histogram (VFH)-Fast Obstacle Avoidance for Mobile Robots, *IEEE Transactions on Robotics and Automation*, 7(3), 1991

[3] D. Fox, W. Burgard, and S. Thrun, The Dynamic Window Approach to Collision Avoidance, *IEEE Transactions on Robotics and Automation*, 4:1, 1997

[4] S. Quinlan and O. Khatib, Elastic Bands:Connecting Path Planning and Control, *IEEE Transactions on Robotics and Automation*, vol 2, Atlanta, USA, 1993, pp. 802-807

[5] J. Minguez and L. Montano, Nearness Diagram Navigation (ND): A new Real Time Collision Avoidacne Appraarch, *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Takamatsu, Japan, 2000.

[6] J. Minguez and L. Montano, Nearness Diagram Navigation (ND): Collision Avoidance in Troublesome Scenarios, *IEEE Transactions on Robotics and Automation*, vol. 20, 2004, pp 45-59.

[7] J. Minguez, J. Osuna and L. Montano, A "Divide and Conquer" Strategy based on Situations to achieve Reactive Collision Avoidance in Troublesome Scenarios, *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, New Orleans, USA, 2004.

[8] B. Hamner, S. Singh, and S. Scherer, Learning Obstacle Avoidance Parameters from Operator Behavior, *Special Issue on Machine Learning Based Robotics in Unstructured Environments, Journal of Field Robotics*, Vol. 23, No. 11/12, December, 2006, pp. 1037-1058.

[9] J. Durham and F. Bullo, Smooth Nearness-Diagram Navigation, *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, 2008.