Multi-behaviors coordination controller design with enzymatic numerical P systems for robots

Xueyuan Wang [a,b], Gexiang Zhang [b,c], Xiantai Gou* [b], Prithwineel Paul [b], Ferrante Neri [d,1], Haina Rong [b] Qiang Yang [e] and Hua Zhang [a]

[a] *School of Information Engineering, Southwest University of Science and Technology, MianYang 621010, China*

[b] *School of Electrical Engineering, Southwest Jiaotong University, Chengdu 611756, China*

[c] *College of Information Science and Technology, Chengdu University of Technology, Chengdu 610059, China*

[d] *COL Labloratory, School of Computer Science, University of Nottingham, Nottingham, United Kingdom*

[e] *College of Control Engineering, Chengdu University of Information Technology, Chengdu 610225, China*

Abstract

Membrane computing models are parallel and distributed natural computing models. These models are often referred to as P systems. This paper proposes a novel multi-behaviors co-ordination controller model using enzymatic numerical P systems for autonomous mobile robots navigation in unknown environments. An environment classifier is constructed to identify different environment patterns in the maze-like environment and the multi-behavior co-ordination controller is constructed to coordinate the behaviors of the robots in different environments. Eleven sensory prototypes of local environments are presented to design the environment classifier, which needs to memorize only rough information, for solving the problems of poor obstacle clearance and sensor noise. A switching control strategy and multi-behaviors coordinator are developed without detailed environmental knowledge and heavy computation burden, for avoiding the local minimum traps or oscillation problems and adapt to the unknown environments. Also, a serial behaviors control law is constructed on the basis of Lyapunov stability theory aiming at the specialized environment, for realizing stable navigation and avoiding actuator saturation. Moreover, both environment classifier and multi-behavior coordination controller are amenable to the addition of new environment models or new behaviors due to the modularity of the hierarchical architecture of P systems. The simulation of wheeled mobile robots shows the effectiveness of this approach.

Keywords. Membrane computing, reactive navigation, autonomous mobile robot, behaviors coordination.

1. Introduction

P systems (PS) are bio-inspired parallel distributed computing models [40,48]. Many variants of P systems have been introduced, inspired by biological phenomena such as the functioning and inter-cellular communication of cells and neurons [50,38]. The computing power and complexity aspects of these models have been studied extensively [2,11,26,36,58]. Moreover, membrane computing models with parallel distributive architecture and membrane creation, deletion and division operations can generate exponential workspace and these variants can solve computationally hard problems, i.e., the **NP**-complete, **PSPACE**-complete problems in polynomial time or even in linear time [2,11,44,37].

In recent years the use of the membrane computing models to solve many real-life problems has also gained interest, especially to solve engineering problems [34]. Some variants such as spiking neural P systems [39,66] have been used for fault diagnosis of the power systems [70,71,52] and image processing [53]. Spiking neural P systems are an important paradigm of Membrane Computing that combines spiking neural networks [17,18,6,16,21]. Also numerical P systems

[1]School of Electrical Engineering, Southwest Jiaotong University, Chengdu, China, Email: 491098063@qq.com; School of Computer Science, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, United Kingdom, Email: ferrante.neri@nottingham.ac.uk.

have been used in robotics [8,41,61] and tissue P systems have been used in image segmentation [10,45]. Another important application of P Systems is to solve optimization problems [69]. Some important studies are in parameter optimization problems in manufacturing [68] and combinatorial optimization problems [71].

Among the many variants of P Systems, Numerical P Systems (NPS) [49] and Enzymatic Numerical P Systems (ENPS) [61] are amongst the most successful due to their high performance in robots' control applications [7,8,41,63,13], especially for modular and complex tasks of autonomous mobile robots (AMR), see also [12,42,65,47,3]. The success of NPS and ENPS in robotics is due to the inherent parallel and distributed nature of these P systems along with their powerful numerical computation power [43].

One of the most fundamental problems in robotics is to obtain a path for the robot from the starting point to the goal [59,62]. When a robot moves in a complex and unknown environment, it faces many possibilities. To reach the goal by overcoming various difficulties is the main challenge. These problems can be solved by recognizing the environment patterns, planning a path, positioning and executing the navigation safely and efficiently [55,64,72,29]. The concept of controllers based on numerical P systems was introduced in [8] and has been further discussed in [63] to design controllers of autonomous mobile robots using ENPS and to solve simple navigation tasks. This work investigates the navigation of the robots in more complex environments by means of controllers based on ENPS. We aim at studying ENPS controllers for autonomous robots which can identify multiple environment prototypes and coordinate the behaviors of the robots within them.

In this study, an environment classifier and a novel multi-behaviors control approach based on ENPS are proposed to enhance the reactive navigation performance of the AMR. The novelty of this approach is mainly in three aspects: (1) 11 prototypes of comprehensive topological maps describing the local environments are considered together to design the classifier for environment identification module; (2) A multi-behavior coordination membrane controller (MBCMC) is presented for behavior coordinator module; (3) A serial control algorithm is developed to guide AMR to avoid obstacle, tend to target and follow a wall, etc.

In order to reduce the error impact of sensor noise and poor obstacle clearance, the membrane classifier is designed based on the "binarized rough model" to produce the precisely desired environment pattern, which is used as the input of the behavior coordinator module. Behavior coordinator uses an enzymatic numerical P system to integrate specific behaviors by a well-thought out local path planning (i.e., path planning in an unknown or partially unknown environment) strategy, without large memory size and heavy computation burden. The specific behavior control algorithm is designed based on the Lyapunov stability theory to produce the precisely desired velocity. Furthermore, the effectiveness of the introduced control approach is verified by applying the simulated AMR.

The remainder of this article is organized as follows. Section 2 describes Multi-Behaviors Dynamic Selection Problems (MBDSP). In Section 3, we depict the proposed behavior based membrane controller in detail for solving MBDSP of AMR reactive navigation. Section 4 presents simulation results. Conclusions are drawn in Section 5.

2. Multi-Behaviors Dynamic Selection Problems and ENPS

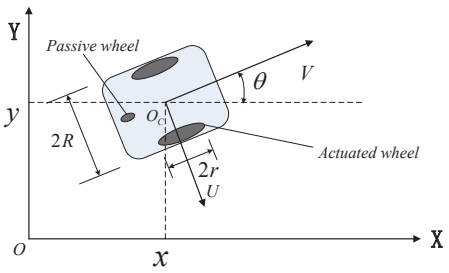
The autonomous robots are capable of self-judgment and independent navigation in an unknown environment. We describe the AMR mechanical system, and MBDSP in the following sections.

2.1. AMR description and Problem Statement

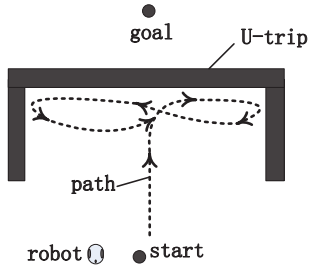
In this study, the AMR mechanical system schematic graph, which is shown in Figure 1(a) consists of two actuated wheels and a back unpowered universal wheel. The passive wheel does not affect the degree of freedom of the kinematic model, and can work with the nonholonomic constraints as follows:

$$\dot{y} \cdot \cos\theta + \dot{x} \cdot \sin\theta = 0 \tag{1}$$

The posture of AMR in global coordinates frame *XOY* is represented by using the Cartesian coordinate vectors with three degrees of freedom $p = \{x, y, \theta\}^T$.



(a) mobile robot schematic



(b) AMR getting trapped in a U-shape trip

Figure 1. AMR description and getting trapped in U-shape trip

The positive direction of θ is anti-clockwise, which is used to guide the angle of a robot. The motion posture of AMR is determined by linear velocity v and angular velocity ω, which is denoted by vector $V = (v, \omega)^T$. Note that, the two wheels are driven by independent torques from two DC motors, where the radius of two wheels are represented by r, while the distance between two driving wheels is denoted by $2R$. It is assumed that the AMR mass center is located at O_c and mounted with non-deformable wheels. The kinematic model for AMR can be represented as in (2) below, where v_r and v_l are the linear velocities of the left and right wheel, respectively, see [51].

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} (v_r + v_l) * \cos\theta/2 \\ (v_r + v_l) * \sin\theta/2 \\ (v_r - v_l)/2R \end{bmatrix} \tag{2}$$

Now we discuss what Multi-Behaviors Dynamic Selection Problems (MBDSP) are. Let us imagine that a robot wants to reach some destination in an unknown environment. At first, the robot follows the planned path and will avoid if some obstacle is blocking the path. If the obstacle is very large, it may decide to walk along the periphery of the obstacle. So there can be many unknown situations in front of the robot and it must have the ability to handle the movements safely and effectively. Hence a group of distinct behavior modes is supposed to help the robot to co-ordinate at each time instant. This is the so-called *Action Selection Problem* in robotic reactive navigation [46], which we have referred to as the MBDSP. The reactive navigation is one of the most challenging problems in AMR. The behavior-based systems are proved to be very responsive to an unknown environment, and the performance of reactive navigation greatly relies on its behavior selection mechanism module. Moreover, there are several aspects about MBDSP which should be paid more attention to

(1) **Behavior control law model:** current controllers usually implement processing of sense-plan-action separately, and do not consider the unity kinematic control law model of different behaviors deliberately, while robots need to wander free not only in maze but also in outdoor and indoor unknown environment;

(2) **Control architecture mode:** current action based architecture is not clear about designing an architecture which allows the dynamic switching among different types of behavior (such as reactive or reflective behavior) selection strategies;

(3) **Multi-behaviors coordination mode:** AMR can very easily fall into the local minima trap when reactive navigation has no prior knowledge of the complex environment. It is also likely to be caused by the first two factors. But an excellent coordinator prevents from these faults. Hence, the dynamic switching strategy, subdivision of different types of behaviors and designing of the corresponding control law are introduced. Furthermore, the behaviors that are usually needed for AMR to wander free in an unknown environment (including outdoor, indoor and maze) are defined clearly in the following:

* Environment classification;
* Path tracking;
* Goal reaching;
* Obstacle avoidance;
* Wall following;
* Corridor walking;
* Emergency U-turn;

* Self rotation;
* ...;

2.2. Related Work

The world's first intelligent mobile robot Shakey [20] was developed at Stanford in 1960's. Following these methods, more and more advanced modern control approaches have been proposed and successfully applied to AMR in industrial contexts [63]. These control approaches can be classified into the following categories according to different control theories:

(1) artificial potential field (APF) [23],
(2) vector field histogram methods [30],
(3) virtual target approaches [67],
(4) dynamic window approaches [19],
(5) fuzzy logic control (FLC) [25],
(6) neural network methods [56],
(7) bug methods [28], and many others. Among the various local or reactive navigation methods, some problems continue to bother them, such as local minimum trap, complex scenarios, lack of prior knowledge, etc.

The well-known traditional APF [23] and its extended methods [24,25,72] are suitable for underlying on-line control in dynamic environments and low processing needs, but it has a problem of local minima [72], which needs to resort global knowledge of the environment at a higher layer. The Bug family methods [15,28] are inspired by bug's behavior on crawling along the obstacle. These approaches are well known for local navigation with minimum sensor, and also for shorter timing, shorter path planning, a simpler algorithm and better performance. But the performance of these approaches depends on the shape of the obstacles in the environment and need some global visual information. Moreover, the Bug family algorithm usually ignores robotic's practical setting (e.g., for kinematic or dynamic constraints). FLC is indeed one of the most fundamental methods and widely used to coordinate numerous basic tasks involved in path planning of behavior-based robots. Many FLC approaches with other complementary techniques were developed to solve some of mobile robot navigation problems in obstacle avoidance [25,27], path tracking [4] and behavior coordination [22,60]. Although FLC rules offer possible implementations of human knowledge and experience which do not require a precise analytical model of the environment, they cannot obtain the optimal solution and mostly fail while dealing with trap situations and complex scenarios [31].

AMR behavior based reactive navigation usually involves many aspects such as environment identification, control structure, dynamic behavior selection strategies, robot physical setting, etc. The study of MBDSP [31,32,33,35,51] usually emphasizes on one or two aspects and the other properties are simplified or ignored.

In this study, most of them are carefully considered to obtain the desired behaviors of the corresponding environment models and reduce the influence of the local minimum traps of complex unknown environments. Unlike APF and bug family methods [15,24,72], which do not care about the robotic physical characteristics completely. But in this paper, the kinematic behaviors are considered to be designed by Lyapunov theory in accordance to robotic characteristics which are suitable for indoor and outdoor environments. Design of the specialized behaviors control law is beneficial for multi behaviors co-ordination. This study also uses an enzymatic numerical P system to improve the parallel computation performance of the environment classifier and behavior coordinator. Thus, the computations are flexible and are in accordance with reactive navigation. For analogy, some studies emphasising the advantages models of parallelisation are reported in [57,54,1].

2.3. Enzymatic numerical P systems

ENPS are naturally distributed and parallel computing models, in which numerical variables store information. Also a set of evolving rules in each membrane region can iterate simultaneously according to the activation conditions, and transmit information between the nodes (membranes). A standard ENPS is as follows [61]:

$$\Pi = (m, H, \mu, (Var_1, E_1, Pr_1, Var_1(0)), \ldots, (Var_m, E_m, Pr_m, Var_m(0))) \quad (3)$$

where

1. m is the number of membranes, $m \geq 1$;
2. H is an alphabet that contains m symbols;

3. μ is a membrane structure;
4. Var_i is the set of variables from membrane i, and $Var_i(0)$ is the initial values for these variables;
5. E_i is a set of enzyme variables from membrane i, i.e., $E_i \subset Var_i$;
6. Pr_i is the set of programs (rules) in membrane i, composed of a production function and a repartition protocol, which have the following two forms.

- Enzymatic form: the jth program $Pr_{j,i} = (F_{j,i}(x_{1,i},...,x_{k_i,i}), e_{t,i}, c_{j,1}|v_1 + ... + c_{j,n_i}|v_{n_i})$, where $e_{t,i} \in E_i$, $F_{j,i}(x_{1,i},...,x_{k_i,i})$ is the production function; k_i is the number of variables in membranes i; $c_{j,1}|v_1 + ... + c_{j,n_i}|v_{n_i}$ is the repartition protocol; n_i is the number of variables contained in membranes i plus the number of variables contained in children and parent membrane of i; the value $q = \frac{F_{j,i}(x_{1,i},...,x_{k_i,i})}{\sum_{n=1}^{n_i} c_{j,n}}$, denotes "unitary portion" to be distributed to variables $v_1,...v_{n_i}$, where these variables can be calculated according to their corresponding coefficients $c_{j,1},...,c_{j,n_i}$ at time t;
- Non-enzymatic form, which is just like the standard NPS: $\Pr_{j,i} = (F_{j,i}(x_{1,i},...,x_{k_i,i}), c_{j,1}|v_1 + ... + c_{j,n_i}|v_{n_i})$.

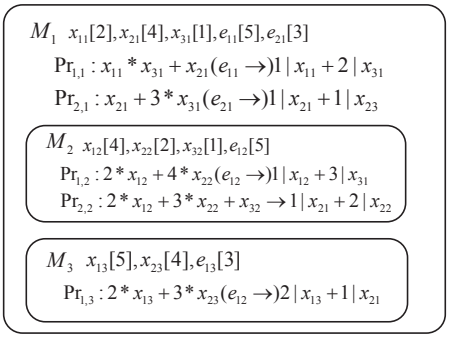


Figure 2. A membrane with an enzyme variable

Inspired by the catalyzing reactions of the biological enzymes, the enzymatic action in ENPS model is to select the valid rules. Here we illustrate how the ENPS works in Figure 2,where there are four variables $x_{11}[2]$,$x_{21}[4]$,$x_{31}[1]$ and $e_{11}[5]$, one production function $x_{11} * x_{31} + x_{21}(e_{11} \rightarrow)$ and one repartition protocol $1|x_{11} + 2|x_{31}$ inside the membrane M_1. In this case, $e_{11} > \min(x_{11}, x_{21}, x_{31})$ and the amount of enzyme is more than the number of variables, which indicates that reaction can take place. Then, function $F_{j,i}(x_{1,i},...,x_{k_i,i}) = 2 * 1 + 4 = 6$ is computed, followed by the the sum of these repartition coefficients calculation: $C_{j,i} = \sum_{n=1}^{n_i} c_{j,n} = 2 + 1 = 3$. The value $q = \frac{F_{j,i}(x_{1,i},...,x_{k_i,i})}{C_{j,i}} = \frac{6}{3} = 2$, denotes "unitary portion" to be distributed to variables $v_1,...v_{n_i}$, where these variables can be calculated according to their corresponding coefficients $c_{j,1},...,c_{j,n_i}$ at time $t+1$. So in this case the new value is $x_{11} = q * 1 = 2$ and $x_{31} = q * 2 = 4$.

ENPS have flexible computing feature. Because of the hierarchical membrane structure with multiple rules in one region characteristics, enzyme variables can be used for conditional transmembrane transport and decide on the rules of evolution direction. The active rules are performed simultaneously inside their membranes, but unnecessary rules are not carried out and the results are distributed in globally uniform way. The computing power of the ENPS, and efficiency of the membrane structure representation for designing robotic behaviors have been investigated in [61] and [8], respectively.

3. Design of Environment Classifier and Behavior Coordination Controller

3.1. Design of Environment Classifier

In order to respond according to the appropriate behavior, AMR should know the relationship between its current status and the local environment at first. The output of the environment prototype will work as the features of the essential environment for navigation, and need not store or deal with unnecessary details.

3.1.1. Local environment prototype:

Based on our understanding of the outdoor or indoor navigation, there are ten cases [56] for a robot, such as: following a left-side wall, wandering in open area, crossing a corridor or meeting a right-side obstacle, etc. Figure 3 lists these ten cases. At the first row of Figure 3 five following cases have been shown: left

wall (LW), right wall (RW), hallway wall (HW), left corner wall (LC) and right corner wall (RC). The five cases of meeting an obstacle are defined at the second row of Figure 3, i.e., front wall (FW), left side (LS), right side (RS), two side (TS) and dead end (DE).

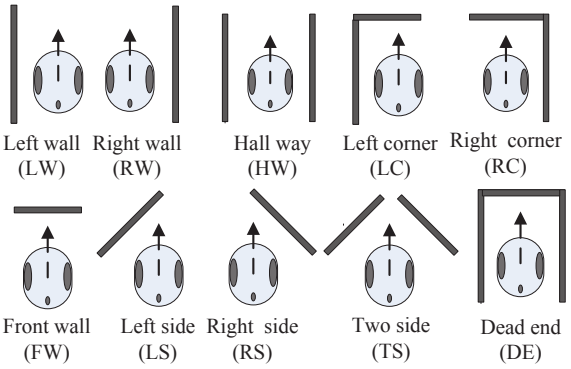


Figure 3. Ten prototypes/cases of local environment robots may meet

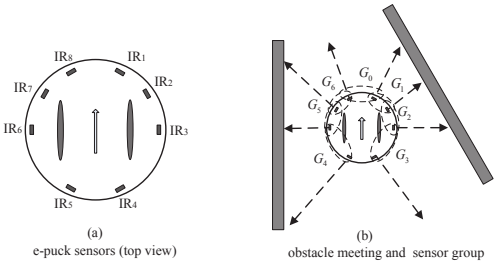


Figure 4. IR sensors placement for the e-puck

Before classification of the various local environments by sensor, the robot's sensor feature must be defined. In order to reduce the cost of a sensor device, e-puck has only eight 8 Infra-red(IR) distance sensor around the body in Figure 4(a). The Figure 4(b) shows the sensors $IR_{1\cdots8}$ layout and the probing direction from the top of the robot. The values from the 8 IR are grouped into $(G_0, G_1, \ldots, G_6)$ as they meet some obstacle or follow some wall. For instance in Figure 4(b), the values for the groups (G_4, G_5) and (G_1, G_2) will be bigger than the other groups when they meet the left wall and right side obstacle conditions (bigger value means smaller distance to obstacle), respectively. Figure 5 shows the 11 sensory patterns registered for the entire prototype environment which correspond to the 10 cases of maximum possibility according to the assumptions in Figure 3 and *NO* represents there is no obstacle in the environment.

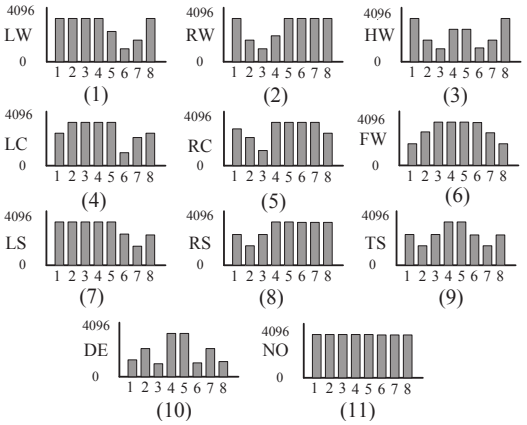


Figure 5. Sensory patterns for 11 cases

3.1.2. Environment classifier design based on ENPS:

In this paper, we propose a local environment classifier based on ENPS to quickly identify the sensory patterns when AMR is surrounded by obstacles. Fast and accurate environment classification is beneficial for the response to the appropriate behavior.

As shown in Figure 6, the environment classifier is designed by using a membrane system with a hierarchical membrane structure containing four membranes. The inner membrane *Compute Environment Model* is used to match the 11 case environment model. According to the sensor data, it has 11 variables, where $s_j \left[\left(sensor_j - p_j^i \right)^2 \right]$, $(j = 1 \cdots 8)$ represent the 8 infrared sensor match errors.

The p_j^i $(i = 1 \cdots 11)$ represent the 11 cases of environment patterns in Figure 5. The enzyme $E_c[v_{in}]$ has the threshold input value v_{in} as the initial value, and it is used to decide whether the rules $\mathrm{Pr}_{i\_j}, sensor_j$ should be executed according to the values of the variable of $s_{1\cdots8}$.

Rule $\mathrm{Pr}_{i\_j}, sensor_j$ is executed when $sensor_j$ is matched with i-th environment pattern $patt_{i\_j}$ success-

fully and the variables $sum_i[0]$ and $Sum_{all}[0]$ are assigned with value 1 simultaneously.

Again, sum_i is used to store the number of successful match of the i_{th} pattern $(i = 1 \cdots 11)$, where larger value represents higher match degree. Then, the numbers are sorted from big to small. The Variable Sum_{all} is proposed to store the total number of successful matches in 11 sensory patterns, which is further used to understand the accelerated sorting instead of traditional sorting method (such as Bubble Sorting, Hill sorting, etc).

The inner membrane *Find Out Several Possible Pattern* is designed to find out several more likely patterns with nine variables, where

(1) $S_{aver}[0]$ is the set of average time of total successful match through rule $\mathrm{Pr}_1, pattern_i$ (i.e., first program for pattern $i = 1, 2, \ldots, 11$).

(2) The variable $pat_i[0]$ represents the distance difference between $pat_i[0]$ and $S_{aver}[0]$ though the rule $\mathrm{Pr}_2, pattern_i$.

(3) The Enzyme $E_{aver}[0]$ is combined with pat_i in $\mathrm{Pr}_3, pattern_i$ to verify whether this rule is applicable or not.

The execution of this rule means, this sensory pattern is a matching environment prototype and the next rules are applicable. The Enzyme $E_{max}[0]$ is set to 9 in $\mathrm{Pr}_3, pattern_i$. Since the pattern variable sum_i must be less than 9, the rule $\mathrm{Pr}_4, pattern_i$ can be applied, and the enzyme $E_{pat\_i}[0]$, $E_{patt\_i}[0]$ are set to pattern value sum_i. The pattern sum variable $M_{sum}[0]$ also accumulate one copy of sum_i. Then the rule $\mathrm{Pr}_5, pattern_i$ is executed and the initial value 1 of variable $num_i[1]$ accumulate to the sum variable $Num_{sum}[0]$ which works as a counter.

The innermost membrane *Find Out Optimal Model* has two variables. The average variable $S_{pat\_i}[0]$ is assigned to the number of the group pattern whose values are bigger than S_{aver} in membrane *Find Out Several Possible Pattern*. So, S_{pat_i} must be larger than S_{aver}, and it can decide whether rule Pr_2 can be activated while combined with the enzyme E_{patt_i}. It can also find out the optimal pattern and the output of the most possible result in the i-th pattern is stored into No. Note that, the enzyme EH in skin membrane *Output Environment Model No* must be assigned to double value of i_{th}. For instance, if the most possible pattern happens at $i = 1$ and EH only get one part value of i_{th}, then $\mathrm{Pr}_2, main$ in skin membrane cannot be activated because of the initial value of variable C_T being 1, and the computing cannot be finished. It is used to ensure that the rule $\mathrm{Pr}_2, main$ in the skin membrane must be executed and the computing is terminated. Meanwhile, the variable Out_{no} in the skin membrane collects the output result of the computation.

3.2. Dynamic Multi-Behavior Coordination

In order to explore complex and unknown environments, AMR not only needs to be promptly reactive, it also must act safely and smoothly. Moreover, AMR can break away from local minima trap and arrive at the goal finally. This section describes how to coordinate with these behaviors by dynamic selection mechanism. It should be noted that the control law design for all behaviors in this article is described in detail in Appendix.

3.2.1. Multi-behavior coordination strategy

The proposed flow chart of dynamic multi-behavior selection is depicted in Figure 7. In Figure 7, $Flag = 1$ means AMR is moving towards the goal until some "obstacles" are detected, where d_{gr} is the distance between goal and robot. It is defined as $d_{gr} = \sqrt{(x_g - x_r)^2 + (y_g - y_r)^2}$, where (x_g, y_g) and (x_r, y_r) represents the coordinate of goal and robot, respectively. It should be getting smaller and smaller while running towards to the goal, but in contrast, if it is becoming bigger, it means that the obstacle avoidance or wall following mode is operated and the robot has moved far away. AMR can determine the accurate status relationship between itself and the obstacle by environment classifier at once.

The "obstacles" can be grouped as

(1) obstacle cases, i.e., FW, LS, RS,

(2) wall follow cases, i.e., LW, RW, HW, LC, RC (corridor walking also classified as this case), and

(3) the dead end cases, i.e., TS, DE.

AMR might fall into the trap while avoiding the obstacle or following the wall. In order to resolve the local minimum problem, AMR must solve the problems such as positional relationship among goal, obstacle, wall and robot. Also must investigate whether the distance d_{gr} is minimal and what kind of obstacle

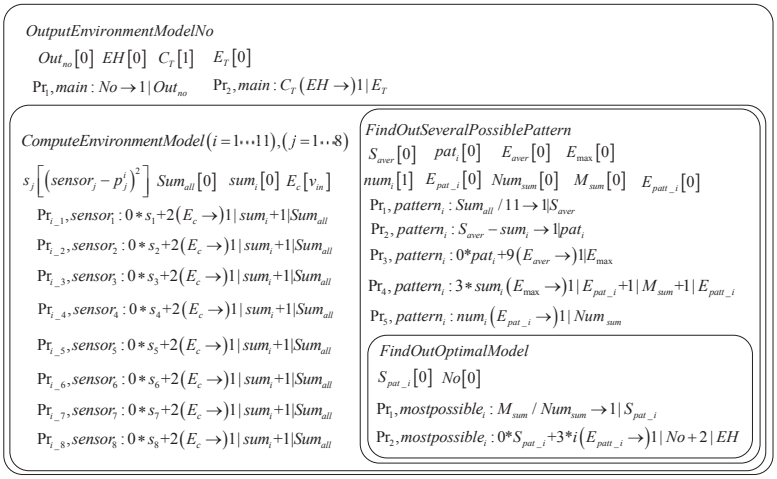


Figure 6. Membrane classifier for 11 environment patterns

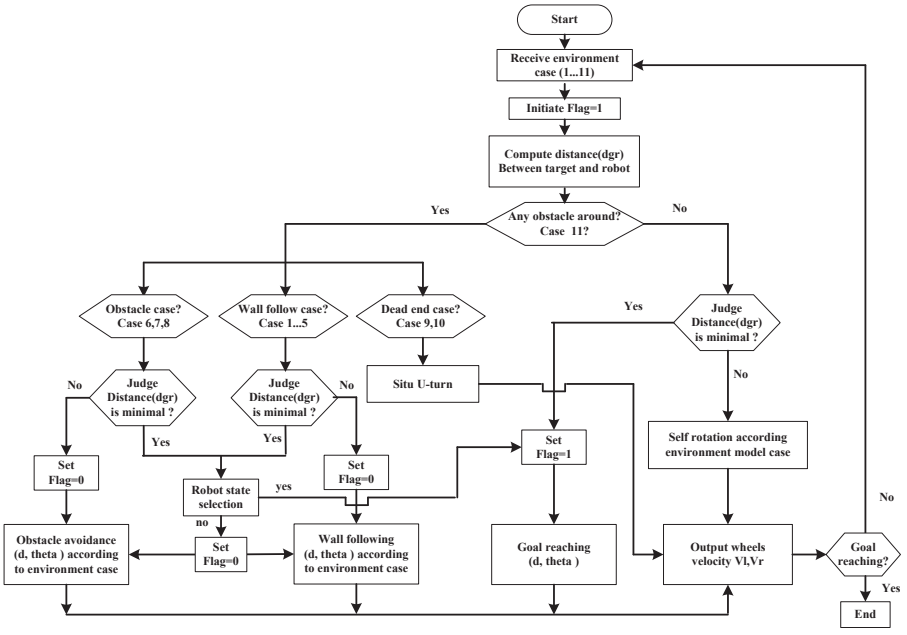


Figure 7. Flow chart of multi behavior coordination controller

is around? For instance, if some obstacles or walls are located at the right side of the AMR according to the environment model case, then the goal is located at the left side of the robot, and d_{gr} is the minimal distance. Also, AMR should enter the goal reaching mode. On the other side, if the goal and obstacle are located on the same side of AMR, then even if d_{gr} is minimal, the goal reaching mode cannot be activated. In another example, in order to go out of the maze, if the robot has just passed the wall (obstacle) and entered into the

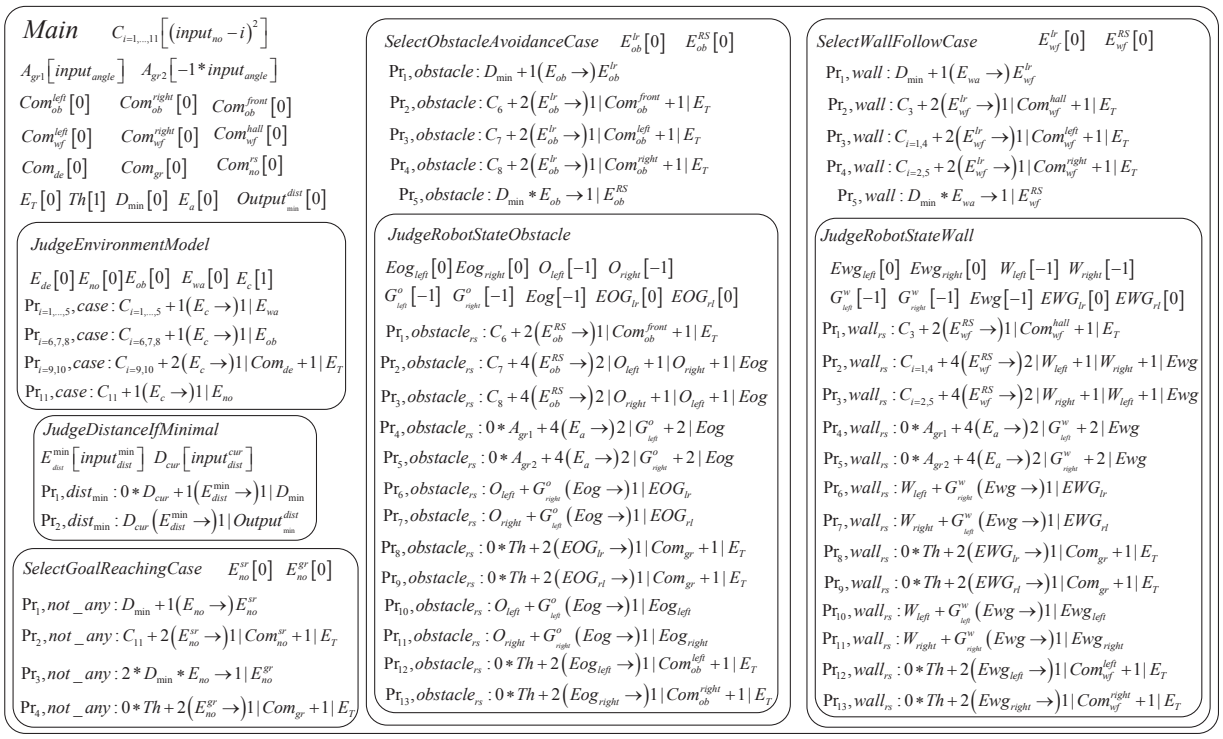


Figure 8. Dynamic multi-behavior coordination membrane controller

open area, it should go to the judge state and select goal reaching mode directly or self-rotation mode to follow wall again.

In this work, an interesting dynamic multi-behavior selection strategy is constructed to speed up the behavior coordination by parallel processing. Moreover, the corresponding co-ordination controller based on P system is shown in Figure 8.

3.2.2. Multi-behavior coordination membrane controller design:

MBCMC is shown in Figure 8. It is designed by using a P system with a hierarchical membrane structure containing eight membranes. The skin membrane *Main* has 27 variables.

(1) $C_i = \left[(input_{no} - i)^2\right]$, $i = 1, 2, \ldots, 11$ are the environment case variables. These variables have the initial value $(input_{no} - i)^2$, $input_{no}$ and it is one of the 11 patterns from environment model membrane classifiers.

(2) $A_{gr1}\left[input_{angle}\right]$ and $A_{gr2}\left[-1 * input_{angle}\right]$ are the angle variables and have the input value $\theta = \theta_R - \theta_G$ as initial value, which depicts the positional relationship between robot and goal. $\theta < 0$ means that the goal is located at the left hand of robot and vice versa (Figure 17(b)).

(3) $Com_{ob}^{left}[0]$,$Com_{ob}^{right}[0]$ and $Com_{ob}^{front}[0]$ are left, right side and front obstacle avoidance behavior control output variable, respectively. Again, $Com_{wf}^{left}[0]$, $Com_{wf}^{right}[0]$ and $Com_{wf}^{hall}[0]$ are left, right wall following and hall crossing behavior computation output variable, respectively.

(4) The variable $Com_{de}[0]$ is the U-turn output variable for dead end case and $Com_{gr}[0]$ is the goal reaching output variable. Moreover, $Com_{no}^{rs}[0]$ is the self-rotation behavior control output variable.

All of the output variables have the initial value 0, but when some of the behavior gets triggered, then the corresponding variable value is set to 1.

(5) The variable $Th[1]$ is the threshold variable with initial value 1 and $D_{\min}[0]$ is the minimal distance variable with initial value 0.

(6) The enzyme variable $E_a[0]$ and $E_T[0]$ have the initial value 0, but when E_T is not equal 0, the controller is terminated.

(7) The $Output_{\min}^{dist}[0]$ is the minimal distance output variable which has initial value 0.

The inner membrane *Judge Environment Model* has five enzyme variables, where

(1) $E_{no}[0]$, $E_{de}[0]$, $E_{ob}[0]$ and $E_{wa}[0]$ work as trigger enzymes for not any obstacle case, meet dead end case, meet obstacle case and wall case, respectively.

(2) Enzyme E_c has initial value 1, and is used to decide whether the 11 rules $\Pr_{i=1,\ldots,11}, case$ should be executed or not according to the environment case variables $C_{i=1,\ldots,11}$. For instance, if $input_{no} = 9$, then the initial value of $C_{11} = (9-11)^2 = 4$ and hence enzyme $E_c < C_{11}$. So, the rule $\Pr_{11}, case$ can not be activated. On the contrary, for rule $\Pr_9, case$, the initial value of $C_9 = 0$, hence $E_c > C_9$, and rule $\Pr_9, case$ is executed. Moreover, Com_{de} has set value 1, U-turn behavior is selected, E_T is set to 1 and controller ends.

The inner membrane *Judge Distance If Minimal* has two variables.

(1) The enzyme $E_{dist}^{\min}$ with the input value $input_{dist}^{\min}$, which is also the minimum distance of all the distances between the robot and goal.

(2) The variable D_{cur} has the input value $input_{dist}^{cur}$ as the initial value, which is the current distance between the robot and goal.

Both of these variables decide whether the rules $\Pr_1, dist_{\min}$ and $\Pr_2, dist_{\min}$ should be applied or not. If $D_{cur} < E_{dist}^{\min}$, both rules are activated and the minimal distance variable $D_{\min}[0]$ is set to 1. Meanwhile, the minimal current distance variable D_{cur} is collected as the output of the variable $Output_{\min}^{dist}[0]$.

The inner membrane *Select Goal Reaching Case* has two enzyme variables, i.e., $E_{no}^{sr}[0]$ and $E_{no}^{gr}[0]$ with initial value 0. This membrane will be activated by enzyme $E_{no} = 1$ as case 11 (no obstacle around the robot).

Rule $\Pr_1, not\_any$ is used to judge the robot. If there is any reverse movement away from the goal, then the value of D_{min} is equal to 0. It means that the robot has just left the obstacle or wall case. The rule $\Pr_2, not\_any$ is activated when $E_{no}^{sr} = 1$ and Com_{no}^{sr} is set to 1. The robot will implement the rotation mode. It will turn left or right according to the previous behavior case. For instance, the robot will turn left when the previous case is left obstacle avoidance or left side wall following, and vice versa. This mode helps the AMR to find the wall or obstacle surface again while running around the maze or to avoid the U shape obstacle. Rule $\Pr_3, not\_any$ will let enzyme E_{no}^{gr} be equal to 1 as $D_{min} = 1$ (trend to goal movement). Again, $E_{no}^{gr} = 1$ will activate the rule $\Pr_4, not\_any$ and the robot will obtain the goal reaching mode. Moreover, E_T is set to 1 and the controller ends.

The inner membrane *Select Obstacle Avoidance Case* has two enzyme variables $E_{ob}^{lr}[0]$ and $E_{ob}^{RS}[0]$ and one sub membrane. This membrane is activated by the enzyme $E_{ob} = 1$ as in case $6, 7, 8$ (front, left or right obstacle).

Rule $\Pr_1, obstacle$ is also used to judge the robot whether there is any reverse movement away from the goal like rule $\Pr_1, not\_any$ in membrane *Select Goal Reaching Case*, and rule $\Pr_2, obstacle$, $\Pr_3, obstacle$ or $\Pr_4, obs\ tacle$ is activated according to the values of C_6, C_7 and C_8. For instance, if $C_8 = 0$, then the rule $\Pr_4, obstacle$ is executed, Com_{ob}^{right} is set to 1 and the robot implements right side obstacle avoidance. Moreover, E_T is set to 1 and the controller ends. On the other side, rule $\Pr_5, obstacle$ assigns $E_{ob}^{RS} = 1$ as $D_{min} = 1$ (trend to goal movement), and it activate the rules in sub membrane *Judge Robot State Obstacle*.

Judge Robot State Obstacle has 5 enzyme variables and 4 common variables.

(1) Enzymes $Eog_{left}[0]$, $Eog_{right}[0]$, $EOG_{lr}[0]$, $EOG_{lr}[0]$, $EOG_{rl}[0]$ have initial value 0 and $Eog[-1]$ has initial value -1.

(2) The common variables $O_{left}[-1]$ and $O_{right}[-1]$ are used to mark the obstacle and locate it at the left or right side of the robot by changing the initial value from -1 to 1.

(3) $G_{left}^{o}[-1]$, $G_{right}^{o}[-1]$ are used to record the goal and locate it at the left or right side of the robot.

The rule $\Pr_1, obstacle_{rs}$ is activated when $C_6 = 0$ (case 6: the obstacle is located at the front of the robot) and both Com_{ob}^{front} and E_T are set to 1.

The AMR should compute the obstacle avoidance at once and without any further analysis. Rule $\Pr_2, obstacle_{rs}$ should be activated as case 7, two contribution is assigned to O_{left}, one contribution is as-

signed to O_{right}, then $O_{left} = -1+2 = 1$, $O_{right} = -1+1 = 0$ (means obstacle is located at left side of the robot). It is same as rule $\mathrm{Pr}_2, obstacle_{rs}$ to represent the right side of the obstacle. Both rules $\mathrm{Pr}_4, obstaclee_{rs}$ and $\mathrm{Pr}_5, obstaclee_{rs}$ are used to judge the location of the goal at the left side or right side of the robot according to the variables A_{gr1} and A_{gr2}.

Enzyme Eog can obtain contribution from rules $\mathrm{Pr}_{2,\ldots,5}, obstaclee_{rs}$. Also if Eog is large enough (2 is obtained in this controller), then it will activate rules $\mathrm{Pr}_6, obstaclee_{rs}$ and $\mathrm{Pr}_7, obstaclee_{rs}$. Moreover, both the rules are used to judge whether the obstacle and goal are located on both sides of the robot.

If rule $\mathrm{Pr}_8, obstaclee_{rs}$ or $\mathrm{Pr}_9, obstaclee_{rs}$ is activated, and both of Com_{gr} and E_T are set to 1, then the AMR should be able to compute the goal reaching. On the contrary, if the rules $\mathrm{Pr}_{10,\ldots,13}, obstaclee_{rs}$ are executed while the obstacle and target are located at the same side of the robot, Com_{ob}^{left} or Com_{ob}^{right} is set to 1. So, AMR continues to maintain obstacle avoidance despite closer to the goal.

The operating mechanism of inner membrane *Select Wall Follow Case* and its sub membrane *Judge Robot State Wall* for wall following are similar to obstacle avoidance. To restrict the length of the paper, we do not expand the description further.

4. Simulation Results

In this section, the performance of the proposed environment model classifier and dynamic multi-behavior co-ordination controller is verified based on the Matlab simulation. Furthermore, the simulation under Webots (robot simulation software) environment is used to test the performance of mobile robot navigations in different environment models. All the simulations are conducted on the PC with CPU 2.8GHz, 4GB RAM, and the software platform MATLAB7.4 and Windows 7 OS. E-puck robot has 8-infrared sensors and Max *IR* value is 4096. The size of the robot is 70 mm in diameter and 55 mm in height with 2 stepper motors.

4.1. Performance metrics

A good selection of the metrics is very important for the control performance. The autonomous robot should reach to the target safely and smoothly in minimum time with the shortest distance. In this section we will introduce some metrics that evaluate the performance of robot motion methods.

1)Time to reach the goal: T_{tog} is the total time to approach to the goal. Less time means better performance.

2)Path length: L_{tog} is the total length of the path from start point to goal point. Shorter length is desirable for better performance. L_{tog} can be represented by equation (4).

$$L_{tog} = \int_{s_0}^{s_n} (1 + (f^{'}(s))^2)^{1/2} ds \tag{4}$$

3)Minimum distance overruns: $D_{obs}^{\min}$ used to measure and record the number of times the sensor value of any channel is less than the minimum safety distance D_{safe} from obstacle. $D_{obs}^{\min}$ can be represented by equation (5).

$$D_{obs}^{\min} = \begin{cases} 0.1 & if\ Min(s_i) \le D_{safe} \\ 0 & otherwise \end{cases} \tag{5}$$

4)Mean distance to obstacles: D_{obs}^{mean} is the mean value of the distance between the obstacle and the robot's sensors in each execution cycle of the entire walking process. Higher values means the walking will be safe.

5)Number of collisions: N_{coll} is the number of times the robot hit an obstacle. The number of the collisions also indicates a degree of safety.

6)Mission failures: N_{fail} is the number of times the mission failed to reach the end. The more times the task fails, the worse the algorithm adaptability.

7)Number of oscillation: T_{osc} is the frequency of change towards the forward direction of the robot during walking, three consecutive switching clocks towards the forward direction are one oscillation behavior, such as left-right-left, or right-left-right. The smaller the value, the smoother will be the walk of the robot. T_{osc} can be obtained by equation (6).

$$Time_{osc} = \begin{cases} 0.1 & if\ the\ robot\ oscillates \\ 0 & otherwise \end{cases} \tag{6}$$

4.2. *Simulation for Environment Classifier*

Since the navigation environment is usually unpredictable, complex and partially unknown, a single environment model membrane classifier can hardly take charge of the whole task. If a single membrane classifier (SMC) is used, it must have a complex structure with many internal parameters to solve the problems of navigation in complex environment. Therefore, a multi-membrane classifier (MMC) (in this paper, two or three) has been employed to identify the environment model with good fault-tolerance capabilities. Since the MMC uses the SMC modules (each covering a specific local environment), it can quickly and easily find good local solutions.

The simulations on e-puck robot with 8 infra-red sensors around have been shown in Figure 4(a). In order to reduce the impact of sensor noise, the sensor's value is filtered with a given threshold before being sent to the membrane classifier. All values smaller than 70 are ignored. At the same time, in order to simplify the environmental identification model, once the value of some sensor is greater than 70 (close to the obstacle), it activates this channel and is set to 1. Otherwise, is set to 0.

Using aforementioned informations, three kinds of SMC can be constructed in the following manner:

$$C_1=\begin{bmatrix}00001110\\01110000\\01111110\\10000111\\11100001\\10000001\\00000111\\11100000\\11000011\\11100111\\00000000\end{bmatrix}\quad C_2=\begin{bmatrix}00000110\\01100000\\01100110\\10000101\\10100001\\10000001\\00000011\\11000000\\01000010\\10100101\\00000000\end{bmatrix}\quad C_3=\begin{bmatrix}00000100\\00010000\\00100100\\10000111\\11100001\\10000001\\00000100\\01000000\\01000010\\10100101\\00000000\end{bmatrix}$$

Figure 9. Binary encoding of C_1, C_2 and C_3 modular

Note that row $1,\ldots,11$ of the binary encoding of C_1, C_2 and C_3 modular represent the 11 environment patterns shown in Figure 5. The last row is all zeros that represents not any obstacle is around the robot. Figure 10 shows that the actual paths are taken by SMC and MMC. SMC uses C_1 and MMC uses C_1 and C_2.

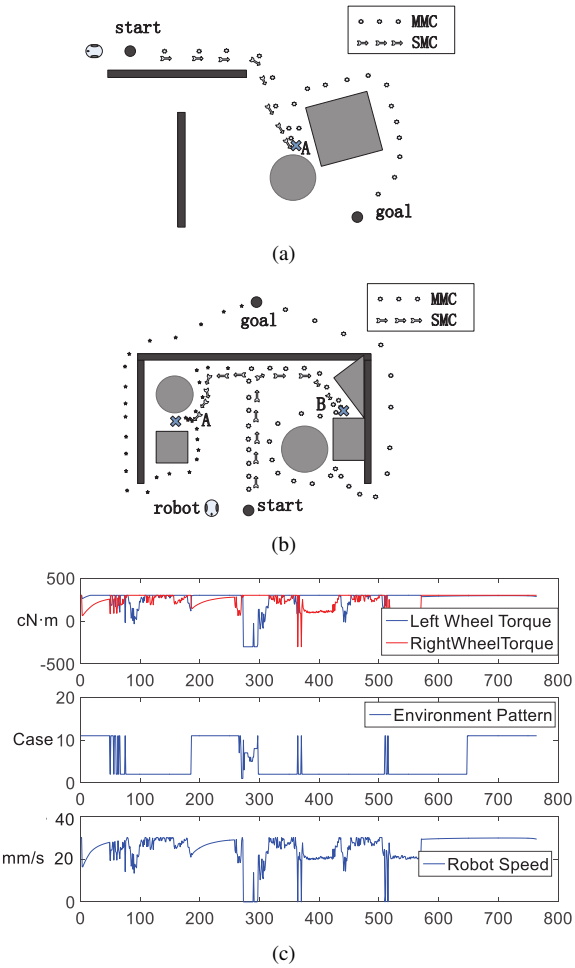


Figure 10. Escape from a local minimum in a complex environment

There are several local minimum traps in Figure 10(a) and (b). As shown in Figure 9, MMC can break away from both of the local minimum trap and arrive at the destination successfully as in Figure 10(a) and (b). But SMC can not struggle to break away from the local minimum trap (A) point in Figure 10(a), and sometimes can not break away from the local minimum trap (A) or (B) in Figure 10(b). SMC alternately judge the environment patterns by switching from case 2 to 7 constantly while reaching the edge of the trap. The environment pattern changes to case 3 (Hall way) while reaching the bottom of the neck trap. But if the robot size is bigger than the spacing of the hall way, it

Table 1. Performance comparisons between SMC and MMC while escape from local minimum

		N_{module}	N_{fail}	T_{tog} (s)	L_{tog} (mm)	$D_{obs}^{\min}$	N_{coll}
Figure 10(a)	SMC	1	10	fail	fail	fail	fail
	MMC	2	0	48.7	355.3	1.3	0
	Single NN [56]	1	10	fail	fail	fail	fail
	M-NN [56]	5	0	49.1	357.6	1.3	0
Figure 10(b)	SMC	1	4	64.2	564.2	1.5	4
	MMC	2	0	57.1	528.3	1.6	1
	MMC	3	0	56.9	526.9	1.7	0

will fall into the trap and the robot speed will become zero while the left and right wheel are still running. But MMC can move away from the trap successfully because MMC judge this pattern as 9 cases and would activate U-turn behavior to break away from the trap. Figure (c) shows the operation-related parameters of the robot when it gets out of the local minimum trap in Figure (b) and reaches the target, where left wheel torque and right wheel torque are the left and right wheel driving torques of the differential robot.

Under the same obstacle configuration as in Figure 10, we have changed the start and goal positions and ran the simulation ten times. For the same navigation task, Table. 1 is listed in the performance of SMC and MMC. The N_{module} is the number of modular and N_{fail} is the number of failures. Whenever SMC fall into the trap in Figure 10(a) every time, N_{fail} is 10. But MMC can have better identification of the environment and can move away from the trap successfully, where N_{fail} is 0 and T_{tog} is the total execution time. Since MMC has a low elapsed time, it has better performance than SMC. The L_{tog} is the total length of the path and SMC has the longer path length than MMC because it walks a duplicate path due to the environment model identify error. The number of collisions N_{coll} indicates a safe navigation. The results in Table.2 show that MMC has a better performance than SMC. In addition, unlike in [56] where "5-by-1" modular neural network (M-NN) environment classifier is required to replace single NN classifier to realize successful navigation in Figure 10(a). This paper considers only two kinds of modular (C_1, C_2) to achieve the same task. As shown in Table.2, the performance of different sizes of $MMC(2,3)$ for Figure 10(b) is not obvious. So, the number of modules used depends on the specific local environment. Furthermore, NN environment classifiers need larger and greater amount of samples to train the controller. There is a need for about 3000 ultrasonic patterns to train NN classifiers [56] and $50,000$ samples with speed of 4.5 hours for training the navigation reservoirs [5]. On the contrary, SMC or MMC based on ENPS does not need to train any processing and is simple to initialize the environment model.

4.3. Simulation for Multi-Behavior Coordinator

Several behavior coordination schemes are employed to evaluate the performance, such as fuzzy logic approach [35], expert fuzzy cognitive map (FCM)-based approach [31], fuzzy discrete event systems FDES-based approach [22], optimized modular NN approach [56]. Throughout the simulations, we have adopted modular C_1 as in MMC in Test I, (C_1, C_2) as in Test II and Test III, modular (C_1,...,C_3) as MMC in Test IV. The cruising speed of the robot is set to 30 mm/sec and the minimum safety distance $D_{safe} = 24mm$. The translation cycle of robot is set to 50 ms. The following simulation tests are carried out for validating the proposed approach.

Test I: **Unstructured environment:** Figure 11 shows navigation of different trajectories by the proposed MBCMC and NN controller [56]. In the environment of Figure11, multiple obstacles of different shapes and sizes are randomly distributed, and set up three different groups in an unstructured environment with the same obstacle layout. Contrast experiments between the starting point and the target point are shown in Figure11 (a)-(f). Considering the randomness of the autonomous robots in exploring the unknown external environment, under the same starting point and target point conditions, the autonomous robot performs computer simulations of ten navigation respectively, and the average of each performance metrics in ten experiments is used as the navigation performance. Table.2 shows the comparison results of the performance metrics of the three groups of exper-

iments. It can be seen that the MBCMC method has better performance than the NN approach. MBCMC has a smooth trajectory, less time overhead, and fewer oscillations.

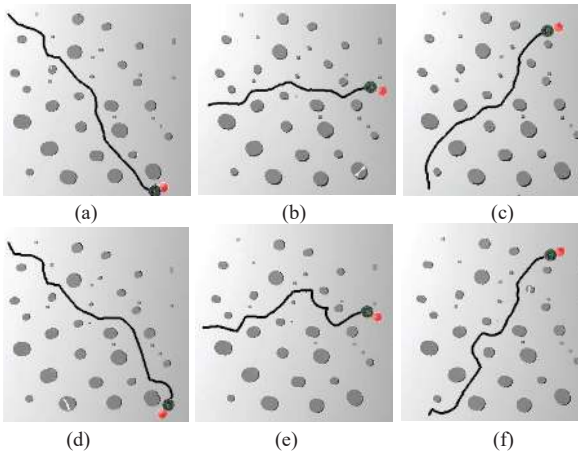


Figure 11. MBCMC (a)-(c) and NN [56] (d)-(f) trajectories in unstructured environment

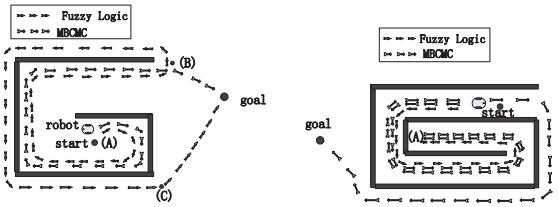
Test II: **G-shape and snail shape environment:** Figure 12 shows the expected results as previously depicted in MBCMC. In [35], a new fuzzy logic controller for robot navigation has been developed, which has adopted an actual-virtual target to escape from the local minimum by defining a sum of turning angles.

If the sum of turning angles throughout the way is near 0^o, the robot would decide to go toward the real target.

If the total amount of turning angles is negative, then the robot will have a counterclockwise motion to compensate the amount at the opening point.

Since the sum of turning angles is -360^o at point "(B)" in Figure 12(a), the robot will not execute goal reaching and will turn counterclockwise to continue following the wall until the point "(B)". But MBCMC will switch the control scheme and run towards the goal directly. Although after breaking away from the G-shape obstacle [35], it will spend more time and run more distance than MBCMC to get goal point.

Snail shape in Figure 12(b) is more complicated trap than G-shape. The distance between the corridors of the snail must be wide enough. The robot in [35] after encountering the first wall (left side or



(a) G-shape (b) snail shape

Figure 12. Escape from a G-shape and snail shape environment

right side), follow the left side or right side wall and then break away from the snail shape obstacle successfully. But the snail shape environment in this paper has a very narrow corridor and with a dead end. Hence, it will effect the definition of the virtual target [35] and event weights of the expert-FCM graph [31]. Also, the robot falls into a trap at dead ends "(A)" as shown in Figure 12(b). Since hall way and dead end are in the general definition of environment patterns and MBCMC can identify those cases in this paper. Moreover, the wall following method is also modified by equation (33) to a suitable corridor environment. The results of Figure 12(b) prove that the robustness of the proposed approach is better than the approaches in [35,31], whether it is a wide corridor or narrow corridor.

Test III: **Building environment:** The robot starts in room 1 and navigates to the goal at room 2. Figure 13 shows that both MBCMC and M-NN [56] started at room 1, crossed the narrow corridor and arrived at the turning point "(A)". MBCMC can implement self-rotation strategy according to the environment model and aim the room 2 as the goal. But the robot (M-NN [56]) failed to enter through the "door" at (A), because it was confused by the corridor module and the left turn module (adopt the competitive coordination). The robot ([56]) can break away the dead end "(B)" in Figure 13, but it spends more time to reach the goal than the proposed approach.

Test IV: **Maze environment:** The performance of MBCMC was examined in the similar environments ([22]) with more complex mazelike traps in Figure 15. Figure 15(b) shows the similar navigation scenarios of the robot moving in the maze environment with irregular obstacles. FDES-based approach [22] employs supervisory control theory of fuzzy discrete event sys-

Table 2. Performance comparison of MBCMC and NN [56] under the same obstacle distribution

		Method	$L_{tog}(mm)$	$T_{tog}(s)$	T_{osc}	D_{obs}^{min}
First group	Figure (a)	MBCMC	636	23.5	1.5	1.2
	Figure (d)	NN in [56]	692	25.4	2.2	1.6
Second group	Figure (b)	MBCMC	552	20.6	1.8	0.7
	Figure (e)	NN in [56]	573	21.2	2.4	0.9
Third group	Figure(c)	MBCMC	688	25.2	1.7	1.1
	Figure(f)	NN in [56]	703	26.3	2.2	1.5

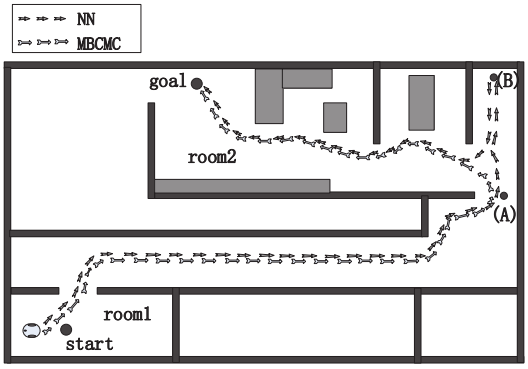


Figure 13. Robot starts at room 1 and goes to room 2

tems to model and control several navigation task of a mobile robot. Two deliberative behaviors ("Go to Target"(*GT*) and "Route Follow" (*RF*)) and three reactive behaviors ("Wall Follow"(*WF*), "Avoid Obstacle"(*AO*) and "Avoid Dead ends"(*AD*)) are weighted through FDES and navigate the robot to the final target successfully. In this method, target seeking is based on following a series of immediate sub-targets (waypoint). *GT* is used for path optimization and aims to find the next nearest waypoint. *RF* is used to navigate the robot through way points. Therefore the robot can trace a collision-free path with optimum distance towards the actual target in maze-like environments. Unlike in [22], the start and end points are identified and moreover the waypoints are given manually. The robot in this paper only knowns the start point and goal point and also can identify the surrounding unknown environments by MMC accurately. The dynamically chosen reasonable behaviors by MBCMC, the AMR can help to walk out of the maze safely. Figure 15 shows the traveled trajectory with these environments, where both MBCMC and FDES-based approach have the similar path. Also, Figure 16 depicts the Yaw angle between robot direction and goal, environment pattern, left and right wheel driving torque and robot speed results of MBCMC related to the complex maze-like environment in Figure 15.

Figure 14 depicts autonomous robot's trajectory in different structured / unstructured maze environments using the proposed MBCMC approach. These different types of maze environments have many different types of local minimum traps. But the robot does not fall into the traps and the trajectory is smoother while maintaining a certain safety distance from obstacles. It shows that MBCMC can well adapt to navigation tasks in different complex environments and find an optimal path to the goal. Table.3 depicts the performance evaluation of the proposed MBCMC with the existing approach in different kinds of environment. Moreover the trajectory is smoother and safer than other methods, the oscillation times of MBCMC is also the smallest. This is due to both the behavior selection strategy and computational efficiency of the membrane controller, as well as the detailed design of each behavior.

5. Conclusions

In this paper, a simple and effective environment pattern membrane classifier is constructed based on parallel distributive computing models known an ENPS. It can be identified by eleven environment patterns and can build or modify environment modules quickly. It is observed that the proposed MMC and MBCMC are able to provide a robust and successful navigation with a smooth path in different type of environments. The proposed bio-inspired controllers are validated on simulated mobile robots and comparison with neural network controller and fuzzy logic controller has been provided The proposed approach eases the design of the behavior-based hybrid control architectures with

Table 3. Performance comparison for the three test environments

Test	Method	$T_{tog}(s)$	$L_{tog}(mm)$	$D_{obs}^{mean}(mm)$	D_{obs}^{min}	N_{coll}	T_{osc}
Test II (G - shape)	Fuzzy logic [35]	58.3	1682	17.9	1.5	2	1.2
	MBCMC	31.1	897	18.5	0.9	0	0.5
Test III (Room)	Neural Network [56]	90.2	2515	14.2	9.1	6	6.3
	MBCMC	85.5	2308	16.8	7.3	0	4.5
Test IV (Maze2)	FDES [22]	108.9	3157	15.8	7.6	0	8.7
	MBCMC	112.7	3278	16.1	6.9	0	6.5

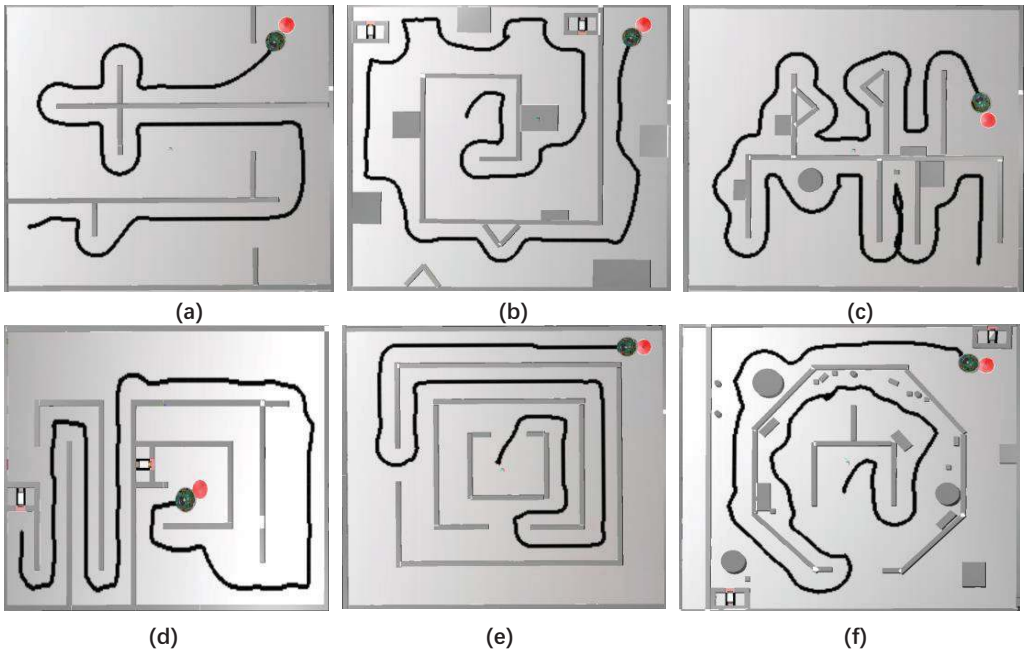


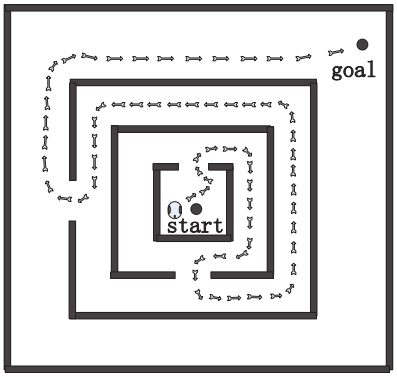
Figure 14. Unstructured environment

the higher modularity which is obtained by associating P systems. Moreover, MMC with binary environment model is able to cope with sensor imprecision and ambiguous situations.

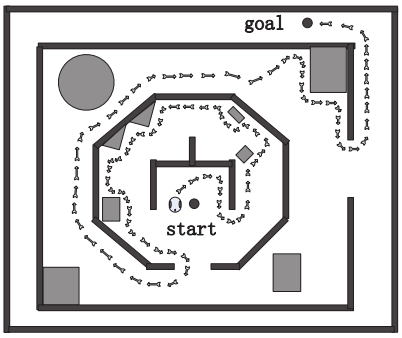
Also, introduction of more behaviors to the membrane hybrid control architecture is easily performed by adding more environment models to MMC and events to MBCMC. To address the more complex navigation tasks, studies on decentralized and modular membrane controller can be carried out in the future.

Acknowledgements

The work is supported by the National Natural Science Foundation of China (61972324, 61672437, 61702428, 61771411), by Beijing Advanced Innovation Center for Intelligent Robots and Systems (2019IRS14), Sichuan Science and Technology Program (2018GZ0086, 2018GZ0185), New Generation Artificial Intelligence Science and Technology Major Project of Sichuan Province (2018GZDZX0043) and Artificial Intelligence Key Laboratory of Sichuan Province (2019RYJ06).

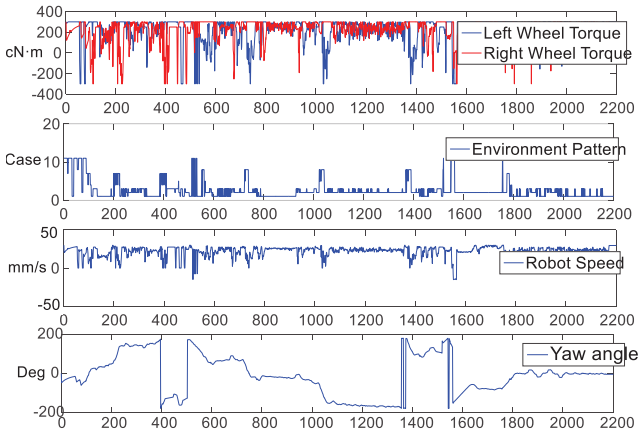


(a) Maze 1

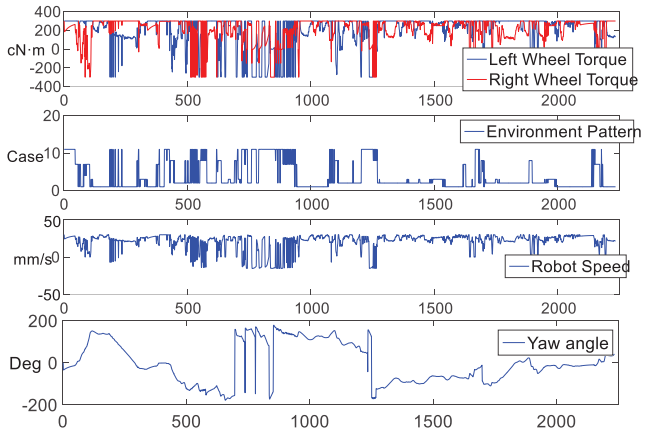


(b) Maze 2

Figure 15. Escape from a maze environment



(a) Control result to maze 1



(b) Control result to maze 2

Figure 16. Control results of MBCMC related to mazelike environment in Figure 15

References

[1] H. Adeli, "High-performance computing for large-scale analysis, optimization, and control," *Journal of Aerospace Engineering*, vol. 13, no. 1, pp. 1–10, 2000.

[2] A. Alhazov, C. Martin-Vide, and L. Pan, "Solving a PSPACE-complete problem by P systems with restricted active membranes," *Fundamenta Informaticae*, vol. 58, no. 2, pp. 66–77, 2003.

[3] M. Almagro-Cádiz, V. Fresno, and F. de la Paz, "Speech gestural interpretation by applying word representations in robotics," *Integrated Computer-Aided Engineering*, vol. 26, no. 1, pp. 97–109, 2019.

[4] G. Antonelli, S. Chiaverini, and G. Fusco, "A fuzzy-logic-based approach for mobile robot path tracking," *IEEE Transactions on Fuzzy Systems*, vol. 15, pp. 211–221, 2007.

[5] E. A. Antonelo and B. Schrauwen, "On learning navigation behaviors for small mobile robots with reservoir computing architectures," *IEEE Transactions on Neural Networks Learning*, vol. 26, no. 4, pp. 763–780, 2015.

[6] M. Bernert and B. Yvert, "An attention-based spiking neural network for unsupervised spike-sorting," *Int. J. Neural Syst.*, vol. 29, no. 8, pp. 1 850 059:1–1 850 059:19, 2019.

[7] C. Buiu and A. George, "Membrane computing models and robot controller design, current results and challenges," *Journal of Membrane Computing*, vol. 1, no. 4, pp. 262–269, 2019.

[8] C. Buiu, C. I. Vasile, and O. Arsene, "Development of membrane controllers for mobile robots," *Information Sciences*, vol. 187, pp. 33–51, 2012.

[9] R. Carelli and E. O. Freire, "Corridor navigation and wall-following stable control for sonar-based mobile robots," *Robotics and Autonomous Systems*, vol. 45, no. 3, pp. 235–247, 2003.

[10] H. A. Christinal, D. D. Pernil, and P. Real, "Region-based segmentation of 2d and 3d images with tissue-like P systems," *Pattern Recognition Letters*, vol. 32, pp. 2206–2212,

2011.

[11] D. Díaz-Pernil, H. A. Christinal, and M. A. Gutiérrez-Naranjo, "Solving the 3-col problem by using tissue P systems without environment and proteins on cells," *Information Sciences*, vol. 430-431, pp. 240–246, 2018.

[12] J. Félez and A. Bermejo, "Design of a counterbalance forklift based on a predictive anti-tip-over controller," *Integrated Computer-Aided Engineering*, vol. 25, no. 3, pp. 273–288, 2018.

[13] A. G. Florea and C. Buiu, "A distributed approach to the control of multi-robot systems using XP colonies," *Integrated Computer-Aided Engineering*, vol. 25, no. 1, pp. 15–29, 2018.

[14] E. Freire, T. Bastos-Filho, M. Sarcinelli-Filho, and R. Carelli, "A new mobile robot control approach via fusion of control signal," *IEEE Transactions on Systems, Man, Cybernetics. Part B: Cybernetics*, vol. 34, pp. 419–429, 2004.

[15] Y. Gabriely and E. Rimon, "Cbug: A quadratically competitive mobile robot navigation algorithm," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1451–1457, 2008.

[16] F. Galán-Prado, A. Morán, J. Font, M. Roca, and J. L. Rosselló, "Compact hardware synthesis of stochastic spiking neural networks," *Int. J. Neural Syst.*, vol. 29, no. 8, pp. 1 950 004:1–1 950 004:13, 2019.

[17] S. Ghosh-Dastidar and H. Adeli, "Improved spiking neural networks for eeg classification and epilepsy and seizure detection," *Integrated Computer-Aided Engineering*, vol. 14, no. 3, pp. 187–212, 2007.

[18] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *Int. J. Neural Syst.*, vol. 19, no. 4, pp. 295–308, 2009.

[19] S. Horn and K. Janschek, "A set-based global dynamic window algorithm for robust and safe mobile robot path planning," in *Proceedings of the 41st International Symposium on Robotics and the 6th German Conference on Robotics Munich, Germany*, 2010, pp. 1–7.

[20] https://www.sri.com/hoi/shakey-the robot.

[21] R. Hu, Q. Huang, H. Wang, J. He, and S. Chang, "Monitor-based spiking recurrent network for the representation of complex dynamic patterns," *Int. J. Neural Syst.*, vol. 29, no. 8, pp. 1 950 006:1–1 950 006:22, 2019.

[22] A. Jayasiri, G.Mann, and R.Gosine, "Behavior coordination of mobile robotics using supervisory control of fuzzy discrete event systems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 41, no. 5, pp. 1224–1238, 2011.

[23] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation,," in *Proceedings of IEEE Conference on Robotics and Automation*, 1991, pp. 1398–1404.

[24] B. Kovács, G. Szayer, F. Tajti, M. Burdelis, and P. Korondi, "A novel potential field method for path planning of mobile robots by adapting animal motion attributes," *Robotics and Autonomous Systems*, vol. 82 (C), pp. 24–34, 2016.

[25] K.Y.Tu and J. Baltes, "Fuzzy potential energy for a map approach to robot navigation," *Robotics and Autonomous Systems*, vol. 54, no. 7, pp. 574–589, 2006.

[26] A. Leporati, L. Manzoni, G. Mauri, A. E. Porreca, and C. Zandron, "Characterizing PSPACE with shallow non-confluent P systems," *Journal of Membrane Computing*, vol. 1, no. 2, pp. 75–84, 2019.

[27] J. Lilly, "Evolution of a negative-rule fuzzy obstacle avoidance controller for an autonomous vehicle," *IEEE Transactions on Fuzzy Systems*, vol. 15, pp. 718–728, 2007.

[28] V. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst obstacles of arbitrary shape," *Algorithmica*, vol. 2, pp. 403–430, 1987.

[29] V. Lumelsky and A. Stepanov, "Dynamic path planning for a mobile automation with limited information on the environment," *IEEE Transactions on Automatic Control*, vol. 31, pp. 1058–1063, 1986.

[30] 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.

[31] O. Motlagh, S. H.Tang, N. Ismail, and A. R. Ramli, "An expert fuzzy cognitive map for reactive navigation of mobile robots," *Fuzzy Sets and Systems*, vol. 201, no. 12, pp. 105–121, 2012.

[32] D. Nakhaeinia and B. Karasfi, "A behavior-based approach for collision avoidance of mobile robots in unknown and dynamic environments," *Journal of Intelligent and Fuzzy Systems*, vol. 24, no. 2, pp. 299–311, 2013.

[33] D. Nakhaeinia, S. Tang, S. M. Noor, and O. Motlagh, "A review of control architectures for autonomous navigation of mobile robots," *International Journal of the Physical Sciences*, vol. 6, pp. 169–174, 2011.

[34] A. Nash and S. Kalvala, "A P system model of swarming and aggregation in a myxobacterial colony," *Journal of Membrane Computing*, vol. 1, no. 2, pp. 103–111, https://doi.org/10.1007/s41965-019-00015-0 2019.

[35] O.Motlagh, S.Tang, and N.Ismail, "Development of a new minimum avoidance system for a behavior-based mobile robot," *Fuzzy Sets and Systems*, vol. 160, no. 13, pp. 1929–1946, 2009.

[36] D. Orellana-Martin, L. Valencia-Cabrera, A. Riscos-Nunez, and M. J. Pérez-Jiménez, "Minimal cooperation as a way to achieve the efficiency in cell-like membrane systems," *Journal of Membrane Computing*, https://doi.org/10.1007/s41965-018-00004-9 2019.

[37] D. Orellana-Martin, L. Valencia-Cabrera, A. Riscos-Nunez, and M. J. Pérez-Jiménez, "P systems with proteins: a new frontier when membrane division disappears,," *Journal of Membrane Computing*, pp. https://doi.org/10.1007/s41 965–018–00 003–w, 2019.

[38] L. Pan, G. Păun, and G. Zhang, "Foreword: starting jmc," *Journal of Membrane Computing*, vol. 1, no. 1, pp. 1–2, 2019.

[39] L. Pan, G. Păun, G. Zhang, and F. Neri, "Spiking neural p systems with communication on request," *International Journal of neural systems*, vol. 27, no. 08, 2017.

[40] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[41] A. Pavel and C. Buiu, "Using enzymatic numerical P systems for modeling mobile robot controllers," *Natural Computing*,

vol. 11, no. 3, pp. 387–393, 2012.

[42] S. Pellegrinelli and N. Pedrocchi, "Estimation of robot execution time for close proximity human-robot collaboration," *Integrated Computer-Aided Engineering*, vol. 25, no. 1, pp. 81–96, 2018.

[43] I. Peréz-Hurtado, M. A. Martínez-Del-Amor, G. Zhang, F. Neri, and M. J. P. Jiménez, "A membrane parallel rapidly-exploring random tree algorithm for robotic motion planning," *Integrated Computer-Aided Engineering*, 2020, to appear.

[44] M. Pérez-Jiménez and A. Riscos-Nunez, "A linear-time solution to the knapsack problem using P systems with active membranes," in *WMC 2003, Springer, Heidelberg*, C. Martin-Vide, G. Mauri, G. Păun, G. Rozenberg, and A. Salomaa, Eds., vol. LNCS, 2933, 2004, pp. 250–268.

[45] D. D. Pernil, M. A. Gutierrez-Naranjo, H. M. Abril, and P. Real, "Designing a new software tool for digital imagery based on P systems," *Natural Computing*, vol. 11, pp. 381–386, 2012.

[46] P. Pirjanian, "Multiple objective behavior-based control," *Robotics and Autonomous Systems*, vol. 31, pp. 53–60, 2000.

[47] A. Prieto, A. Romero, F. Bellas, R. Salgado, and R. J. Duro, "Introducing separable utility regions in a motivational engine for cognitive developmental robotics," *Integrated Computer-Aided Engineering*, vol. 26, no. 1, pp. 3–20, 2019.

[48] G. Păun, *Membrane Computing: An Introduction.* Berlin Heidelberg: Springer-Verlag, 2002.

[49] G. Păun and R. Păun, "Membrane computing and economics: Numerical P systems," *Fundamenta Informaticae*, vol. 73, no. 1, pp. 213–227, 2006.

[50] G. Păun, G. Rozenberg, and A. Salomaa, *The Oxford Handbook of Membrane Computing*. NY, USA: Oxford University Press, 2010.

[51] R.J.Wai, C.M.Liu, and Y.W.Lin, "Design of switching path-planning control for obstacle avoidance of mobile robot," *Journal of The Franklin Institute*, vol. 348, pp. 718–737, 2011.

[52] H. Rong, K. Yi, G. Zhang, J. Dong, P. Paul, and Z. Huang, "Automatic implementation of fuzzy reasoning spiking neural p systems for diagnosing faults in complex power systems," *Complexity*, no. 2635714, 2019, 16 pages.

[53] J. L. Rosselló, V. Canals, A. Oliver, and A. Morro, "Studying the role of synchronized and chaotic spiking neural ensembles in neural information processing," *International Journal of Neural Systems*, vol. 24, no. 5, Article No. 1440003 2014.

[54] K. C. Sarma and H. Adeli, "Data parallel fuzzy genetic algorithm for cost optimization of large space steel structures," *International Journal of Space Structures*, vol. 18, no. 3, pp. 195–205, 2003.

[55] M. Shen, Y. Wang, Y. Jiang, H. Ji, B. Wang, and Z. Huang, "A new positioning method based on multiple ultrasonic sensors for autonomous mobile robot," *Sensors*, vol. 20, no. 1, pp. 237–252, 2019.

[56] S.J.Han and S.Y.Oh, "An optimized modular neural network controller based on environment classification and selective sensor usage for mobile robot reactive navigation," *Neural Computing and Applications*, vol. 17, no. 2, pp. 161–173, 2008.

[57] R. Soegiarso and H. Adeli, "Parallel-vector algorithms for optimization of large steel structures," *Computer-Aided Civil and Infrastructure Engineering*, vol. 13, no. 3, pp. 207–217, 1998.

[58] P. Sosik and A. Rodriguez-Paton, "Membrane computing and complexity theory: A characterization of PSPACE," *Journal of Computer and Systems Sciences*, vol. 73, pp. 137–152, 2007.

[59] M. G. Soto and H. Adeli, "Recent advances in control algorithms for smart structures and machines," *Expert Systems*, vol. 34, no. 2, 2017.

[60] S. Tang, D. Nakhaeinia, and B. Karasfi, *Application of fuzzy logic in mobile robot navigation, fuzzy logic-controls, concepts, theories and applications*, ser. ISBN: 978-953-51-0396-7, 2012.

[61] C. I. Vasile, A. B. Pavel, I. Dumitrache, and G. Păun, "On the power of enzymatic numerical P systems," *Acta Informatica*, vol. 49, no. 6, pp. 395–412, 2012.

[62] N. Wang and H. Adeli, "Algorithms for chattering reduction in system control," *Journal of the Franklin Institute*, vol. 349, no. 8, pp. 2687 – 2703, 2012.

[63] X. Wang, G. Zhang, F. Neri, J. Zhao, M. Gheorghe, F. Ipate, and R. Lefticaru, "Design and implementation of membrane controllers for trajectory tracking of nonholonomic wheeled mobile robots," *Integrated Computer-Aided Engineering*, vol. 23, pp. 15–30, 2016.

[64] L. Wenjun, K. Yu, and Q. Jiahu, "Indoor localization for skid-steering mobile robot by fusing encoder, gyroscope, and magnetometer," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 6, pp. 1241–1253, 2019.

[65] S. Wu, G. Zhang, F. Neri, M. Zhu, T. Jiang, and K. Kuhnert, "A multi-aperture optical flow estimation method for an artificial compound eye," *Integrated Computer-Aided Engineering*, vol. 26, no. 2, pp. 139–157, 2019.

[66] T. Wu, F.-D. Bilbie, A. Păun, L. Pan, and F. Neri, "Simplified and yet turing universal spiking neural p systems with communication on request," *International Journal of Neural Systems*, vol. 28, no. 8, 2018.

[67] X.Yang, M. Moallem, and R.V.Patel, "A layered goal-oriented fuzzy motion planning strategy for mobile robot navigation," *IEEE Transactions on Systems, Man, Cybernetics, PartB: Cybernetics*, vol. 35, no. 6, pp. 1214–1224, (2005).

[68] G. Zhang, J. Cheng, M. Gheorghe, and Q. Meng, "A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems," *Applied Soft Computing*, vol. 13, no. 3, pp. 1528–1542, 2013.

[69] G. Zhang, M. Gheorghe, L. Pan, and M. Pérez-Jiménez, "Evolutionary membrane computing: a comprehensive survey and new results, ," *Information Sciences*, vol. 279, pp. 528–551, 2014.

[70] G. Zhang, M. Pérez-Jiménez, and M. Gheorghe, *Real-life Applications with Membrane Computing (Emergence, Complex-*

ity and Computation). Berlin Germany: Springer, 2017.
[71] G. Zhang, H. Rong, F. Neri, and M. Pérez-Jiménez, "An optimization spiking neural P system for approximately solving combinatorial optimization problems," *International Journal of Neural Systems*, vol. 24, no. 5, Article No. 1440006 2014.
[72] T. Zhang, Y. Zhu, and J. Song, "Real-time motion planning for mobile robots by means of artificial potential field method in unknown environment," *Industrial Robots*, vol. 37, pp. 384–400, 2010.

6. Appendix

6.1. Multi-Behavior Design

In order to adapt to the local environment, AMR reflective behavior and reactive behavior are properly designed according to the physical characteristics of the robot.

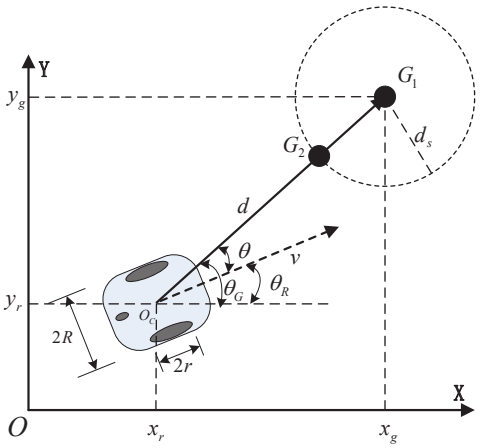


Figure 17. Defining Path tracking and Goal reaching

6.1.1. Goal reaching:

Goal reaching is a behavior that orders the robot to move from the current position to a destination by receiving the desired goal position from the top of the deliberative layer. The description for goal reaching is shown in Figure 17. The current position of AMR with respect to the goal position is expressed in form of the polar coordinates where d represents the distance between goal point G_1 and AMR current point O_c. Again, d_s specifies the safety distance of goal reaching, and θ is the angle error between the current robot heading vector θ_R and goal vector θ_G, $\theta = \theta_R - \theta_G$. Let the robot safety distance and AMR speed limited are as depicted in Figure 17, then the kinematic equations [14] of AMR can be described by the following equations

$$\dot{d} = -v * \cos\theta; \dot{\theta} = -\omega + \frac{v * \sin\theta}{d} \tag{7}$$

where v and ω represent the robot's linear and angular velocities, respectively. First, select the Lyapunov candidate function

$$V_{gr} = \theta^2/2 + \int_0^d S(\tau)\tau d\tau \tag{8}$$

which is a positive definite function and the time derivative of equation (8) is

$$\dot{V}_{gr} = \theta * \dot{\theta} + S(d) * d * \dot{d} \tag{9}$$

When we substitute (7) into (9), we obtain

$$\begin{aligned}\dot{V}_{gr} = \theta * \left(-\omega + \frac{v * \sin\theta}{d}\right) + S(d) * d * (-v * \cos\theta) = \\ = \dot{V}_{gr}^1 + \dot{V}_{gr}^2\end{aligned} \tag{10}$$

The control law of linear velocity is

$$v = v_{\max} * S(d) * \cos\theta \tag{11}$$

where $v_{\max}$ is the maximum of linear speed and S function is defined below

$$S(d) = \begin{cases} 1, & d > d_s \\ 1 - \left(\frac{d_s - d}{d_s}\right)^2, & 0 < d \le d_s \end{cases} \tag{12}$$

The variable d_s in (12) decides the deceleration distance of AMR while reaching the goal. When the robot is far from the goal $(i.e., d > d_s)$, AMR approaches the target as fast as possible, and begins to slow down while reaching the desired target. Substituting (11) into the second partial $\dot{V}_{gr}^2$ of (10), since

$v_{\max} > 0$, $d > 0$, one gets the semidefinite negative function

$$\dot{V}_{gr}^2 = -v_{\max} * d * S^2(d) * \cos^2\theta \leq 0 \qquad (13)$$

Then, substituting (11) into the first partial of (10), $\dot{V}_{gr}^1$ rewriting as

$$\dot{V}_{gr}^1 = \theta * \left(-\omega + \frac{v_{\max} * S(d) * \cos\theta * \sin\theta}{d}\right) \qquad (14)$$

The control law of angular velocity is proposed as

$$\omega = k * \theta + \frac{v_{\max} * S(d) * \cos\theta * \sin\theta}{d} \qquad (15)$$

where k is a proportional constant and $k > 0$. Substituting (15) into (14) which results in another semidefinite negative function $\dot{V}_{gr}^1 = -k * \theta^2 \leq 0$. Hence, one can conclude that the first derivative of the Lyapunov function (16) is the semidefinite negative function, $d = 0$ and $\theta = 0$, which results in $\dot{V}_{gr} = 0$, i.e.,

$$\dot{V}_{gr} = \dot{V}_{gr}^1 + \dot{V}_{gr}^2 \leq 0 \qquad (16)$$

The proposed controller guarantees that $v \leq v_{\max}$ for all $t \geq 0$, and drives the states $d(t)$ and $\theta(t)$ asymptotically to zero. In addition, the goal reaching strategy is based on the Lyapunov function which utilizes the target distance information, has good portability. Moreover, it is inclusive and is applied to design control laws of obstacle avoidance, wall following, corridor walking with a unified "virtual target". In addition, the control laws take into account the safety distance close to the target and the maximum speed of AMR, for which it has better performance of efficiency and safety.

6.1.2. Obstacle avoidance:

This controller is responsible for avoiding the obstacles that may appear randomly when AMR is moving towards the target or following a wall. It is a reflective action and is designed by the goal reaching method described above. When an obstacle is detected, we suppose that a dynamic goal will appear ahead of the robot motion direction to lead it to walk around obstacle smoothly. In Figure 18, an obstacle is around the robot and the environment classifier accurately judge it to locate it on the right side of the robot. Thus, one should set a new "virtual target" on the left side of the robot to make ARM turning left in order to prevent the collision. First, it must select the appropriate sensor IR_i (since several sensors detect the obstacle simultaneously, we must select the suitable ones to define the reference direction of "virtual target"). Since the obstacle is located at the right side of the robot, first find the non-zero value sensors (represent obstacle detection) from the front of left side layout sensor IR_6 to the front of right side ones IR_3 ($IR_6 \rightarrow IR_7 \rightarrow IR_8 \rightarrow IR_1 \rightarrow IR_2 \rightarrow IR_3$). If the obstacle is located at left side of the robot, then find the first non zero sensors from IR_3 to IR_6 ($IR_3 \rightarrow IR_2 \rightarrow IR_1 \rightarrow IR_8 \rightarrow IR_7 \rightarrow IR_6$).

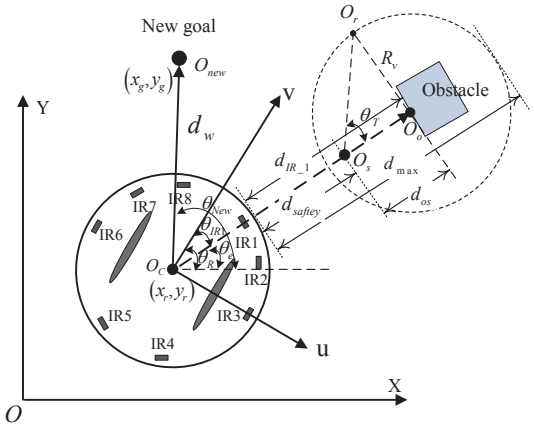


Figure 18. State describe for obstacle avoidance

In this case, IR_1 is found as the candidate sensor, where d_{max} is the maximum measurable distance of sensor, d_{safety} is the safety turning distance of the robot, d_{IR_1} is the distance reading of IR_1. The virtual pressure radius is denoted by R_v and $R_v = d_{\max} - d_{IR\_1}$, d_{os} is the distance between O_o and O_s, where O_o is the contact point from the ray of sensor IR_1 to the surface of the obstacle. In fact, $d_{os} = d_{IR\_1} - d_{safety}$. So, the virtual turn angle of robot θ_T is

$$\theta_T = \arctan\left(\frac{R_v}{d_{os}}\right) = \arctan\left(\frac{d_{\max} - d_{IR\_1}}{d_{IR\_1} - d_{safety}}\right) \qquad (17)$$

The direction of the new goal O_{new} is parallel to the line segment O_rO_s. Thus, the desired new goal position can be given by

$$O_{new} = \begin{bmatrix} x_g \\ y_g \end{bmatrix} = \begin{bmatrix} x_r + k * d_w * \cos\theta_{New} \\ y_r + k * d_w * \sin\theta_{New} \end{bmatrix} \quad (18)$$

where d_w is the desired distance of the new goal ("virtual target"), k is the proportional coefficient and the variable θ_{New} is the orientation of new goal O_{new} attached on the robot platform centroid O_c measured from the horizontal axis (OX).Moreover, θ_{oa}, the obstacle avoidance angle between the current robot heading line and the straight line connecting the points of O_{new} and O_c, is obtained by using the following equation:

$$\begin{aligned} \theta_{oa} &= \theta_{New} - \theta_R = \theta_T - \theta_{IR1} \\ \theta_{New} &= \theta_e + \theta_T \\ \theta_e &= \theta_R - \theta_{IR1} \end{aligned} \quad (19)$$

where θ_R is the orientation of the robot measured from the horizontal axis (OX), and θ_{IR1} is the mounted angle of the sensor IR_1 attached to the robot local coordinate frame (u, O_c, v). Hence, θ_e is equivalent to the angle of the sensor vector (IR_1) relative to the horizontal axis (OX).

Then, the kinematic equations of AMR for obstacle avoidance can also be described by using the following equations

$$\begin{aligned} \dot{d}_w &= -v_{oa} * \cos\theta_{oa}; \\ \dot{\theta}_{oa} &= -\omega_{oa} + \tfrac{v_{oa} * \sin\theta_{oa}}{d_w} \end{aligned} \quad (20)$$

where v_{oa} and ω_{oa} represent the robot's linear and angular velocities for obstacle avoidance, respectively. Select the Lyapunov candidate function

$$V_{oa} = \frac{1}{2}\left(\theta_{oa}^2 + d_w^2\right) \quad (21)$$

Equation (18) defines a local new goal ahead of the robot motion direction, precisely it converts the repulsive force field of the obstacle into a gravitational field of "virtual target". Obviously, as the robot-obstacle distance starts decreasing, θ_T will become bigger and the new goal of the robot is shifted to the opposite direction rapidly. Then, a deviation control signal of angular velocity ω_{oa} and line velocity v_{oa} are generated by goal reaching methods

$$\begin{aligned} v_{oa} &= \kappa_{oa} * d_w * \cos\theta_{oa} \\ \omega_{oa} &= k_{oa} * \theta_{oa} + \kappa_{oa} * \sin\theta_{oa} * \cos\theta_{oa} \end{aligned} \quad (22)$$

where k_{oa} and κ_{oa} are the proportional coefficients. The reaction capability of the controller is regulated by a proper pre-definition of those constants. Hence, one can conclude the derivative of Lyapunov function (21) is the semidefinite negative function as

$$\begin{aligned} \dot{V}_{oa} &= d_w\dot{d}_w + \theta_{oa}\dot{\theta}_{oa} = \\ &= -\kappa_{oa}\cos^2\theta_{oa}d_w^2 - k_{oa}\theta_{oa}^2 \le 0 \end{aligned} \quad (23)$$

In this way, the proposed obstacle avoidance controller can stably avoid obstacles.

6.1.3. Wall following:

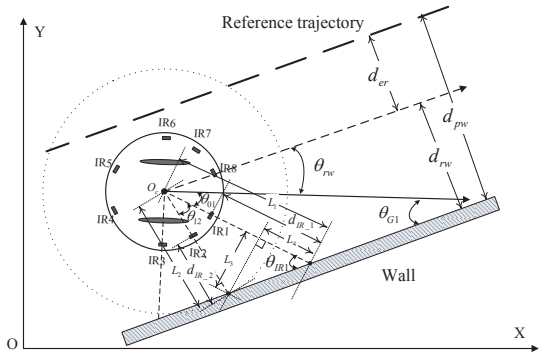


Figure 19. State describe for wall following

Wall following is the robot's ability to follow a wall. These abilities are intended to complement other reflective behaviors in narrow spaces and corridors. The basic objective of the wall following is the generation of a reference trajectory, parallel to a wall (even to a bigger obstacle surrounding). If the distance and angular information between the robot and wall are considered, it can use the goal reaching method to design

the wall following controller. In [14,9], the wall of the surface followed by the robot is supposed to be parallel or perpendicular and the coordinates (X,O,Y) are used in order to simplify the design. Consider a more practical situation as in the wall in Figure 19 which is an arbitrary arrangement. The angle θ_{rw} between the direction of robot motion and the parallel line of the wall can be computed by the angles between the infrared sensors ray and wall surface. In Figure 19, environment classifiers judge the wall located at the right side of the robot. So, several right side group of sensors (G_1,G_2,G_3) in Figure 4(b) are used to compute θ_{rw}. In this case, G_1 is used to get θ_{G1}. For instance, G_1 has two sensors IR_1 and IR_2. The θ_{IR1} is the angle between the vector of IR_1 and wall surface. Moreover, L_1 and L_2 are the line length from the centroid O_c to the wall surface through sensors IR_1 and IR_2, respectively. Again, $L_1 = d_{IR\_1} + R$, $L_2 = d_{IR\_2} + R$, where d_{IR_1} and d_{IR_2} are the reading data of IR_1 and IR_2 and R is the robot's radius. Hence, θ_{IR1} can be given by

$$\begin{aligned} &\theta_{IR1} = \arctan\left(\tfrac{L_3}{L_4}\right); L_3 = L_2 * \sin\theta_{12}; \\ &L_4 = L_1 - L_2 * \cos\theta_{12} \end{aligned} \tag{24}$$

where, θ_{12} is the angle between sensor IR_1 and IR_2, θ_{01} is the angle between the sensor IR_1 and the robot motion direction. So, θ_{G1} is represented by

$$\theta_{G1} = \theta_{IR1} - \theta_{01} \tag{25}$$

The error angle between the robot motion direction and the wall surface is θ_{rw}, and hence we can get

$$\theta_{rw} = \frac{\sum_{i=1}^{n} \theta_{Gi}}{n} \tag{26}$$

where n is the number of effective group sensors, and by effective group it means that both the sensors can be obtained from the distance information, θ_{Gi}, also can be obtained by equation (24) and (25).

In this case, if the assumed AMR walk along the wall at uniform speed v_{wf}, then the kinematic equation is reduced to

$$\dot{d}_{er} = v_{wf} * \sin\theta_{rw}; \dot{\theta}_{rw} = \omega_{wf} \tag{27}$$

where d_{er} is the error distance between the robot centroid O_c and reference trajectory. Select the Lyapunov candidate function for wall following as

$$V_{wf} = \frac{1}{2}\left(d_{er}^2 + \theta_{rw}^2\right) \tag{28}$$

Then, the control law of the angular velocity ω_{wf} is obtained in the following manner

$$\omega_{wf} = -k_{wf} * \theta_{rw} - d_{er} * v_{wf} * \tfrac{\sin\theta_{rw}}{\theta_{rw}} \tag{29}$$

where k_{wf} is the positive proportional coefficients. Again, $d_{er} = d_{pw} - d_{rw}$, d_{pw} is the distance of the given reference trajectory to wall, d_{rw} is the distance from the robot centroid O_c to the right side wall and can be represented by

$$d_{rw} = \frac{\sum_{i=1}^{n} d_{Gi}}{n} + R \tag{30}$$

$$d_{Gi} = \frac{d_{IR\_i1} + d_{IR\_i2}}{2} \tag{31}$$

where d_{IR_i1} and d_{IR_i1} are the distance datas of two sensors in right side effective group G_i, respectively.

Then, the first derivative of Lyapunov function (28) is the semidefinite negative function as

$$\begin{aligned} &\dot{V}_{wf} = d_{er}\dot{d}_{er} + \theta_{rw}\dot{\theta}_{rw} \\ &= d_{er}v_{wf}\sin\theta_{rw} - k_{wf}\theta_{rw}^2 - d_{er}v_{wf}\sin\theta_{rw} \\ &= -k_{wf}\theta_{rw}^2 \le 0 \end{aligned} \tag{32}$$

Thus, the proposed wall following controller can stable trending to reference trajectory.

6.1.4. Corridor walking:

This case is similar to the wall following: the reference trajectory is the middle line of the corridor. The control law can be extended to solve the corridor walking as considered in Figure 3. For this case (i.e., HW), both the left and right side group sensors get the distance data. If $\sum_{i=1}^{4} d_{IR\_i} > \sum_{i=5}^{8} d_{IR\_i}$, it means that the robot is closer to the right side wall and hence it should follow the right side wall. Otherwise, follow the left side wall. In order to let the robot running in the middle of the hallway as far as possible, d_{pw} is reset to

$$d_{pw} = \frac{d_{rw} + d_{lw}}{2} \tag{33}$$

where d_{lw} is the distance from the robot centroid O_c to the left side wall and can be obtained by equation (30) and (31). The Lyapunov candidate function is also treated as equation (28).

6.1.5. Self rotation:

When the robot comes to the corner of the wall, the ending point of the corridor, local trap points, or other similar situations, the robot will have self-rotation in a clockwise/counterclock mode. Suppose the expected rotation angle is θ^*, and the kinematic equations of self rotation is defined as $\dot{\theta}_{sr} = \omega_{sr}$. The deviation of the actual angle from the expected angle and its first derivative are

$$\begin{aligned} \theta_{sr\_d} &= \theta_{sr} - \theta^* \\ \dot{\theta}_{sr\_d} &= \dot{\theta}_{sr} - \dot{\theta}^* = \omega_{sr} \end{aligned} \tag{34}$$

The control law of angular velocity is proposed as

$$\omega_{sr} = -k_{sr}(\theta_{sr} - \theta^*) \tag{35}$$

where k_{sr} is the proportional coeffi-cients. Define the Lyapunov candidate function for self rotation and get its first derivative as following

$$\begin{aligned} V_{sr} &= \tfrac{1}{2}\theta_{sr\_d}^2 \\ \dot{V}_{sr} &= \theta_{sr\_d}\dot{\theta}_{sr\_d} = -k_s r\theta_{sr\_d}^2 \leq 0 \end{aligned} \tag{36}$$

Thus, the control law of self rotation is also stable.

6.1.6. Emergency U-turn:

The emergency U-turn means that the robot should do a U-turn while meeting the environment mode, as in the case of (DE) in Figure 3. It is activated when the distance between the robot and the obstacle becomes smaller than a certain value. Also, after U-turn, the robot continue the navigation module according to the environment classification. U-turn can be treated as a special case of self-rotation, where $\theta^* = \pi$.

6.2. Stability analysis of multi-behavior coordination controllers

To accomplish a goal navigation, it may need to activate one of the multi-behaviors (goal reaching, obstacle avoidance, wall following, corridor walking, self rotation and emergency U-turn) in stages several times. Thus, the Lyapunov candidate function V of the whole system can also be composed of the combination of the six behaviors. Due to the unknown characteristics of the navigation environment, the Lyapunov candidate function V of the whole system cannot be predetermined for a specific navigation task. Nonetheless, according to the conclusions in [14], the proposed phased multi-behavior coordination controllers guarantee can that the robot can always reach the expected position.