Hindawi Publishing Corporation
Journal of Robotics
Volume 2009, Article ID 986207, 13 pages
doi:10.1155/2009/986207

Research Article

Collaborator: A Nonholonomic Multiagent Team for Tasks in a Dynamic Environment

Jing Ren and Mark Green

Faculty of Engineering and Applied Science, University of Ontario Institute of Technology, Oshawa, ON, Canada L1G 7K4

Correspondence should be addressed to Jing Ren, jing.ren@uoit.ca

Received 19 January 2009; Revised 1 September 2009; Accepted 16 October 2009

Recommended by Jorge Manuel Dias

In our previous work, we proposed a potential field-based hybrid path planning scheme for robot navigation that achieves complete coverage in various tasks. This paper is an extension of this work producing a multiagent framework, Collaborator, that integrates a high-level negotiation-based task allocation protocol with a low-level path planning method taking into consideration several real-world robot limitations such as nonholonomic constraints. Specifically, the proposed framework focuses on a class of complex motion planning problems in which robots need to cover the whole workspace, coordinate the accomplishment of a task, and dynamically change their roles to best fit the task. Applications in this class of problems include bomb detection and removal as well as rescuing of survivors from accidents or disasters. We have tested the framework in simulations of several tasks and have shown that Collaborator can satisfy nonholonomic constraints, cooperatively accomplish given tasks in an initially unknown dynamic environment while avoiding collision with other team members. Finally we prove that the proposed control laws are stable using the Lyapunov stability theory.

1. Introduction

Motion planning is a fundamental and important issue in robotics. A special type of path planning includes tasks such as demining [1] and searching for victims after a tragedy, which are extremely crucial jobs and require complete coverage in the process of exploration. If time allows, robot teams should perform a thorough search and cover every square foot of the workspace. While covering too much area is not a concern, insufficient coverage could result in disastrous consequences. Similar criteria apply to many other robotic applications such as vacuum robots and land mine detection. In some more complex scenarios, robots are often interrupted by other tasks. For example, in jobs such as bomb detection and removal [2], or rescuing of survivors from accidents or disasters [3], search is the typical mode. However when a robot discovers an object, such as a survivor, its search phase is paused and then it attends to its rescue task. It can only resume searching after the rescue task is finished. These tasks require some sort of *role change* such as the transformation from rescue to search to successfully complete their mission. Moreover when the task cannot be accomplished by a single robot, coordination or task allocation issues arise. In this instance, robots may negotiate to solve the issues and distribute the tasks among themselves.

The robotics literature contains a number of motion planning strategies for this class of tasks using neural networks, fuzzy logic, approximate cellular decomposition, exact cellular decomposition, and artificial potential fields. Most prior work focuses on either high-level task allocation or low-level task execution. Few works combine task allocation with potential-field based motion planning to provide a holistic solution. Parker [2] provides a behavior-based architecture that facilitates fault tolerance: the effectiveness of the architecture is demonstrated through a team of mobile robots performing a laboratory version of hazardous waste cleanup. Gerkey and Mataric [4] presents an auction-based task allocation, which can achieve a distributed approximation to the global optimum for resource usage. The primary contribution of the work is to empirically demonstrate that the distributed negotiation mechanism is viable and effective for coordinating a physical multirobot team. [5] showed

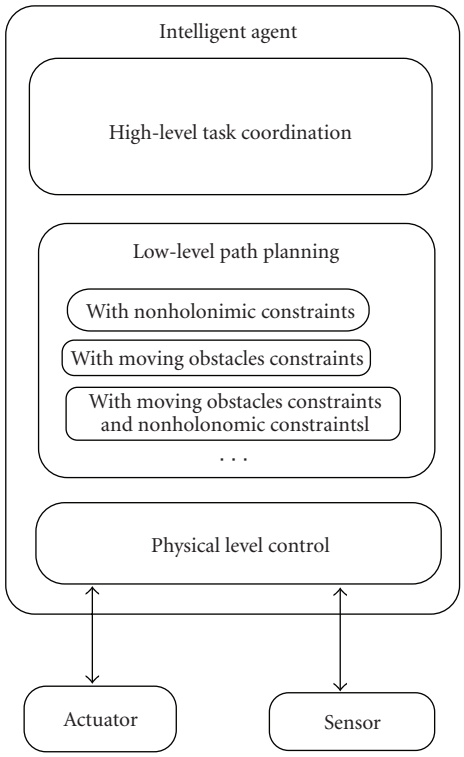


Figure 1: Agent-based navigation software architecture. The architecture of a single agent is structured in four hierarchical layers: *DecisionMaking (DM)*, *KnowledgeBase (KBase)*, *Interaction* and *Communication*. Local and remote sensor information is integrated into the knowledge base, and no distinction is made between them during motion planning.

that the coordination of reactive robots can be obtained through the exploitation of local interactions; however, it basically remains at the level of task allocation and does not propagate to the level of task execution. In our previous work, a novel hybrid navigation scheme is proposed for a multiagent team aiming to bridge the gap between high-level task coordination and low-level path planning [6]. This work did not focus on low-level path planning, assumed ideal working conditions, and did not take into account moving obstacles and nonholonomic constraints. Those aspects of low-level path planning however, are critical to a physical implementation. In a real-life work environment, moving obstacles are often inevitable. For instance, cleaning robots often find themselves in situations where they need to avoid collision with people going about their own business. Moreover, unlike simulated robots, many physical robots can only move forward and backward, subject to nonholonomic constraints, similar to that of vehicles. In this paper, we extend our previous results to a more realistic environment considering each single constraint and both constraints simultaneously.

For a complex system such as the one previously described, the central objective is to achieve system stability. Systems are considered stable when convergence to the global minimum has occurred, which can only be achieved when the entire task is completed. Robots often need to work in an environment which is dynamic, noisy and unpredictable. Nevertheless, they should be reliable in their ability to finish the task despite all of the distractions including the intervention of moving obstacles. This guarantee of convergence is necessary for critical tasks such as rescuing victims after a disaster. Stability for single agent navigation in a static environment can be achieved by constructing a simple Lyapunov function. However, stability for multiagent team navigation in a dynamic environment is inherently more difficult due primarily to the interaction among agents and the presence of moving obstacles. In this paper, we propose a cooperative hybrid navigation scheme and construct a mode-specific team Lyapunov function, which shows that the system is stable at all times by using arguments from hybrid systems theory and Lyapunov stability theory.

The rest of the paper is divided into a number of sections that address individual aspects of this problem. In Section 2, we describe the hierarchical agent design that is the basis for this work. Section 3 presents a potential field-based hybrid navigation scheme. In Section 4, a motion control law is derived and Section 5 extends the framework to environments with moving obstacles. Section 6 incorporates nonholonomic constraints in the agent design, while Section 7 considers the robots with both nonholonomic constraints and moving obstacles. Section 8 presents simulation results for a multirobot system subject to nonholonomic constraints in a dynamic environment. In Section 9 team stability is analyzed. Finally in Section 10, we make concluding remarks and discuss future research directions.

2. Agent Design

The multilayer agent architecture used in this work is depicted in Figure 1. Each of the robots in the team use the same architecture and there is no centralized control for the team. The highest level is the task coordination or knowledge level. At this stage of abstraction, agents analyze and manage tasks using an inherent coordination mechanism. The middle level entails path planning, where agents choose the suitable navigation function to pursue ideal objectives based on the accumulation of knowledge or team decisions. Lastly, the lowest level interacts with an actuator and sensors to govern motion control.

Our agent team is completely decentralized, where there is no leader or coordinator for the team and each agent analyzes and makes navigation and coordination decisions autonomously based on its own knowledge of the world. This design is inherently fault tolerant and easy to reconfigure in the event that task requirement changes [6].

Our previous work describes the agent architecture in detail and how it is used to complete high-level tasks [6]. Here we provide a brief description of its main components and how they assist with low-level path planning. The architecture of a single agent consists of four hierarchical layers: *DecisionMaking (DM)*, *KnowledgeBase (KBase)*, *Interaction* and *Communication*. In the following paragraphs, we define the functionality of each layer as well as a the interface

between the layers. For each of these layers, the control software is multithreaded, with each main object running in its own thread.

The *Decision Making (DM)* layer is the main source of intelligence in the System. It performs navigation planning based on the information stored in the *Knowledge Base* using the hybrid approach described in [6]. At any point in time a robot can be in one of a finite set of modes, which represent high-level navigation goals. A rule-based system is used to transition between the modes based on the current state of the task. The modes define the navigation goals for the low-level path planning system.

The *Knowledge Base* contains the agent's most recent knowledge about the world map, the "group knowledge" as well as the inference rules for coordination. This information includes the location of all known obstacles and any goals that have been located. The information in the *Knowledge Base* is built up from the robot's own experience, plus that of the other robots it is communicating with.

Devices at the *Coordination* layer, discussed in [7] separate the interaction tasks from the main periodic motion planning task. The interaction between agents is performed using an unambiguous communication protocol [7]. We define interfaces between the interaction devices and both the *Knowledge Base* and *Communication* layers. At the *Interaction* layer, devices perform coordination as well as other interaction tasks and store the results in the *Knowledge Base*, without interfering with regular decision making. As a result, it is a simple matter to add new interaction devices if necessary for new applications.

In summary, the higher levels of the architecture provide the locations of the obstacles and goals which are the input to the low-level path planning system. The higher level ensures that the whole space is searched and all the intended tasked are completed, while the lower level deals with navigating robots from their current locations to goal locations.

3. Potential Field-Based Navigation Scheme

A local minimum free potential field can be constructed using generalized spherical potential functions or harmonic functions [8]. In this paper we use a two-dimensional generalized Gaussian function which is much easier to construct and more effective for obstacles whose protective areas can be superscribed by circles without overlapping.

Attractive objects centered at (a_x, a_y) are represented by the negative Gaussian *attractor* function:

$$f_A(x,y) = 1 - e^{-((x-a_x)^2+(y-a_y)^2)/2\sigma^2}. \quad (1)$$

Repulsive obstacles and other agents centered at (r_x, r_y) are modeled with the circular, two-dimensional generalized Gaussian *repulsor* functions:

$$f_R(x,y) = e^{-(1/2)(((x-r_x)^2+(y-r_y)^2)/\sigma^2)^C}. \quad (2)$$

For some positive integer C. The variance, σ, is a measure of the size of the obstacle. The parameter C determines the *effective range (steepness)* of the obstacle and can be varied to modify the obstacle's repulsive effect. We represent obstacle shapes with a superscribed circle.

By specifying a minimum clearance between obstacles and highly localizing the influence of the obstacles through modifications to the parameter C, generalized Gaussian function modeling can be used to construct potential fields that are free of undesired local minima. Figure 2(a) uses a small C and there are local minima among the obstacles, by changing C to a larger value and thereby highly localizing the effects of the obstacles, local minima disappear in Figure 2(b).

In [6], we proposed a mode-switching technique, where mode switches are based on events such as unsearched sectors decreasing and the number of found objects/obstacles increasing. Using the technique of artificial potential fields, we construct a *navigation function*, $V_i^{X_i}(q)$ for agent i in mode X_i (where X_i can be any of the modes defined above). To construct the navigation function for a given mode, we use the three part formula:

$$V_i^{X_i}(q) = VA^{X_i}(q_i) + VO^{X_i}(q_i) + \sum_{j \neq i} VR\left(q_i, q_j\right). \quad (3)$$

In (3), $VA^{X_i}(q_i)$ represents the sum of the effects on agent i of all the N_A attractors in the system during mode X_i, and thus

$$VA^{X_i}(q_i) = \sum_{k=1}^{N_A} \left(1 - e^{-(((x_i-(a_k)_x)^2+(y_i-(a_k)_y)^2)/2\sigma^2}\right). \quad (4)$$

$VO^{X_i}(q_i)$ represents the sum of the effects on agent i of all the known static obstacles in the system during mode X_i. The number of known static obstacles is N_O; so we can state this as

$$VO^{X_i}(q_i) = \sum_{k=1}^{N_O} \left(e^{-(1/2)(((x_i-(r_k)_x)^2+(y_i-(r_k)_y)^2)/\sigma^2)^C}\right). \quad (5)$$

Finally, the functions $VR(q_i, q_j)$ represent the repulsor functions between pairs of agents i and j. Note that VR is not mode dependent, since the number of agents is assumed to be constant. Thus, in general

$$VR\left(q_i, q_j\right) = e^{-(1/2)(((x_i-x_j)^2+(y_i-y_j)^2)/\sigma^2)^C}. \quad (6)$$

4. Motion Control Law

Using the technique of artificial potential fields, we construct a *navigation function*, $V_i^{X_i}(q)$ for each agent i in mode X_i (where X_i can be any of the modes defined above). The (mode-dependent) kinematics of each agent are then given by:

$$\dot{q}_i = -\alpha \frac{\partial V_i^{X_i}/\partial q_i}{\left|\partial V_i^{X_i}/\partial q_i\right|}. \quad (7)$$

In (7), the operator $\partial V_i^{X_i}(q)/\partial q_i$ represents the gradient of $V_i^{X_i}(q)$ with respect to only q_i. Thus, we use a gradient descent method to generate a unit vector direction for $\dot{q}_i$ and

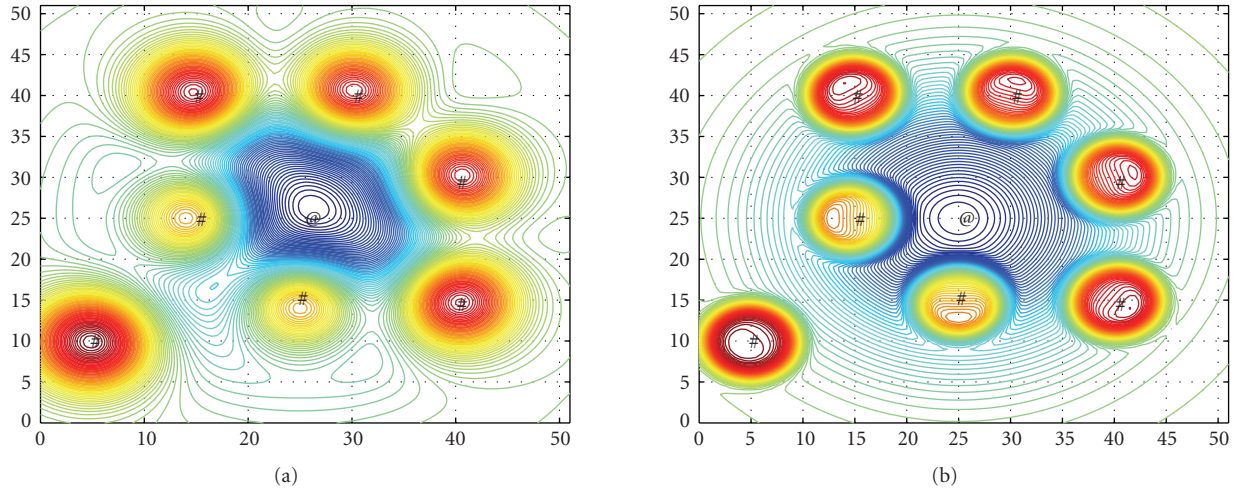


Figure 2: Effect of parameter C.

the constant velocity parameter α to determine agent speed. We use a unit gradient since our Gaussian-like potential functions decay very rapidly.

Note that the gradient decent method gives the direction of motion towards the goal, but the speed of this motion, represented by α, is a function of the robot design. That is, each robot has a maximum speed that must be accounted for by the navigation system.

5. Extension to Environments with Moving Obstacles

Moving obstacle avoidance is of great importance in robotics research and is inherently harder than dealing with only static obstacles. Many works have been devoted to this problem [9–12]. The most related work in literature is presented by Esposito and Kumar in [12]. In that work, Esposito and Kumar propose a nonlinear programming method for computing optimal feasible directions for the mobile robots. By moving along feasible directions instead of the negative gradient direction, robots can successfully avoid all the moving obstacles while retaining a stable trajectory. In other words, the optimal direction decreases the potential function value most rapidly within the feasible direction set. It is not optimal in term of time or shortest distance though. Although that method is effective in many cases, it may fail to converge for some steps, especially in cases where the number of moving obstacles increases (i.e., more nonlinear constraints need to be considered). Also this method is computationally expensive due to the iterations in the optimization process (please refer to [13]).

Our algorithm uses geometry transformation to solve the moving obstacle avoidance problem. Similar to the work of Esposito and Kumar, this algorithm achieves moving obstacle avoidance by finding optimal feasible solutions. However, there is no approximation in the proposed method and only several geometry transformations are involved even for the most complex scenarios; so it can achieve accurate solutions with less computation. More importantly, the proposed method can guarantee to find an optimal solution when there is one.

Another advantage of the proposed method is that it is easier to incorporate the algorithm into various motion planning algorithms. For example, we will show in Section 7 that the proposed algorithm can be easily incorporated into the nonholonomic robot's motion planning [14] and achieve moving obstacle avoidance.

We associate with each moving obstacle a dynamic constraint [12] of the following form:

$$g_j(q) = \left| q_i - P_j \right|^2 - D_{\text{obs}}^2 \geq 0, \tag{8}$$

where P_j is the position of the moving obstacle j, and D_{obs} is the minimum safe distance between the robot and the moving obstacle.

When moving obstacles are far away from robots, they are ignored by the planning algorithm. When the robot moves within the threshold distance D_{obs} of a moving obstacle, the constraints become activated, and robots move along the feasible motion directions. If the path is totally blocked by obstacles (static and moving), the robot will simply halt until it is cleared, which is the same strategy that is used in [15].

In the context of moving obstacle avoidance, *feasible motion directions* denoted by d_{fi} are the directions that can stabilize the system and move robots away from the activated moving obstacles and thus satisfy the following conditions:

$$d_{\text{fi}}^T d_{\text{ngi}} > 0, \tag{9}$$

$$d_{\text{fi}}^T dg_j \geq 0, \quad j = 1, 2, \ldots, m, \tag{10}$$

where $dg_j = \partial g_j / \partial q_i$ represents the gradient of the constraint g_j and $d_{\text{ngi}} = -(\partial V_i^{X_i}(q)/\partial q_i)/\|\partial V_i^{X_i}(q)/\partial q_i\|$ is the negative gradient direction of the navigation function $V_i^{X_i}(q)$ of robot i. In (9), we require any control which can stabilize the system to have a positive projection on d_{ngi} (moving towards

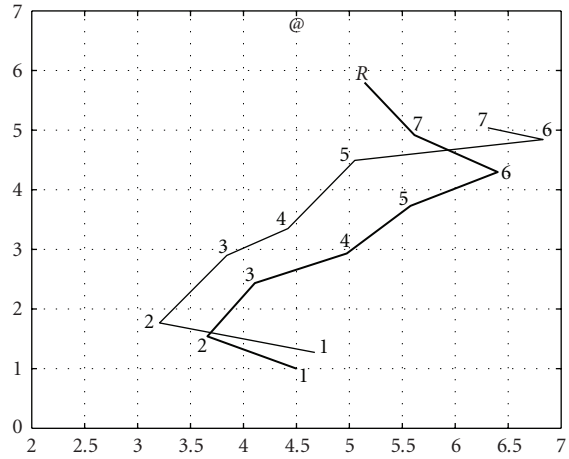


Figure 3: The darker line is the trajectory of a robot trying to reach the goal (@ at the top of the figure). It must achieve this goal while avoiding the moving obstacle (lighter line).

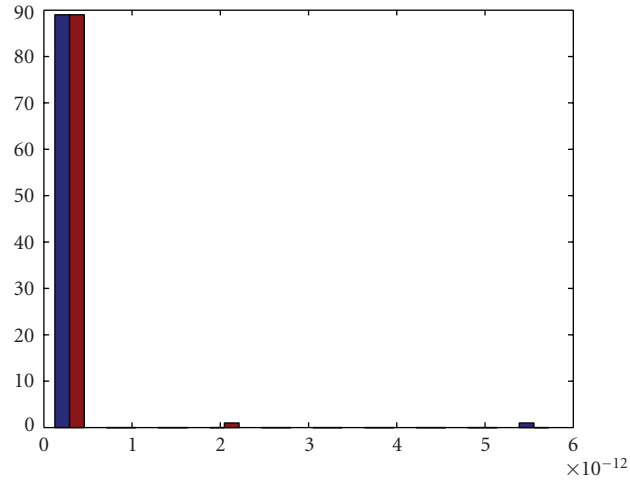


Figure 4: Histogram of position differences of 350 steps in both x and y directions shows that the trajectories generated from two algorithms are the same.

the goal). In (10) we require any control which can move the robot away from the moving obstacle to have a positive projection on dg_j (moving away from the moving obstacles).

Theorem 1. *The feasible motion direction set can be constructed only using the robot gradients of the navigation function and the constraints for moving obstacles. (Please see the appendix for the proof.)*

In our setting, the robot will move along the optimal feasible direction calculated from the constructed feasible set, where the direction is optimal in the sense that it is closest to the negative gradient direction. We will show later in this section that the resulting motion directions are the same as the optimization method in [12] when the optimization method converges. Recall that the set $[\Theta_{\min}, \Theta_{\max}]$ is the set of feasible directions in the following form of a rotation angle with respect to the negative gradient direction d_{ngi}. From basic geometry (refer to Figure 13), we can see that there will be no feasible solution if $\Theta_{\min} > \Theta_{\max}$. If there exists a feasible solution and d_{ngi} is in the feasible set, then d_{ngi} is the optimal feasible gradient direction, otherwise the optimal feasible gradient direction can be obtained by rotating d_{ngi} to the feasible direction set with smallest rotation.

If we use Θ_{ri} to denote the optimal rotation angle, the following formula for Θ_{ri} is given by

$$\Theta_{\text{ri}} = \begin{cases} 0 & \text{if } \Theta_{\min} \le 0,\ \Theta_{\max} \ge 0, \\ \Theta_{\min} & \text{if } \Theta_{\min} > 0, \\ \Theta_{\max} & \text{if } \Theta_{\max} < 0. \end{cases} \tag{11}$$

We define the optimal feasible gradient direction d_{fngi} to be

$$d_{\text{fngi}} = d_{\text{ngi}} B_{\text{ri}}^{T}(q), \tag{12}$$

where the rotation matrix $B_{\text{ri}}(q)$ is of the following form:

$$B_{\text{ri}}(q) = \begin{bmatrix} \cos\Theta_{\text{ri}} & -\sin\Theta_{\text{ri}} \\ \sin\Theta_{\text{ri}} & \cos\Theta_{\text{ri}} \end{bmatrix}. \tag{13}$$

The control law is then given by

$$\dot{q} = k \cdot d_{\text{fngi}}. \tag{14}$$

In Figure 3, we illustrate this algorithm in a typical goal reaching task. The robot is required to reach the goal while avoiding the moving obstacle.

In Figure 4, we show that the results obtained from this method are the same as those from optimization method in [12] when the optimization method converges. Figure 4 shows the histogram of the trajectory differences between optimization method and the proposed method in both x and y direction, we can see the differences are the order of 10^{-15}, which is not significant.

The proposed algorithm is much more computationally efficient than the optimization method. We use moving obstacles with randomly generated positions and run the program 50 times (totally 350 steps) for the simple scenario illustrated in Figure 3, the average number of iterations is above 5 for optimization method, and we know that when it comes to some complex scenarios, the number of iterations will increase substantially. The proposed algorithm, however, always has the computation complexity of $O(1)$, so it is at least five times faster than the optimization method on average.

6. Nonholonomic Robots

The nonholonomic robots we consider in this paper are the Hilare type mobile robot, which have the most typical nonslip constraint. This robot has two parallel wheels which can be controlled independently. By commanding the same

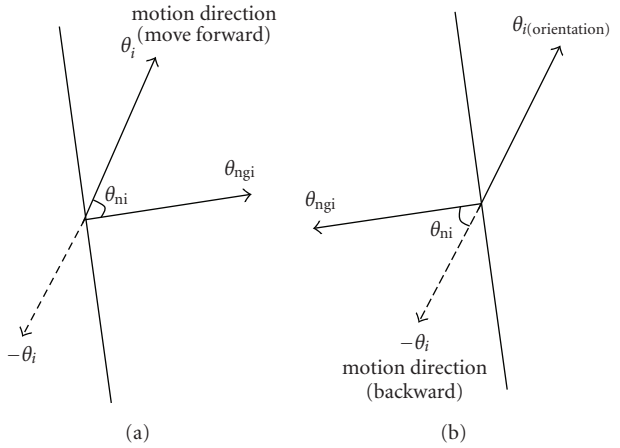


Figure 5: (a) The robot forward direction has a positive projection on the negative gradient direction; the robot will move forward to guarantee the θ_{ni} is in the set $[-\pi/2, \pi/2]$. (b) The robot backward direction has a positive projection on the negative gradient direction; the robot will move backward to guarantee that the θ_{ni} is in the set $[-\pi/2, \pi/2]$.

velocity to both wheels, the robot moves in a straight line. By commanding velocities with the same magnitude but opposite directions, the robot pivots about its axis. The nonslip constraint forces the mobile robot to move only forward or backward. We resolve the constraint by selecting the forward or backward direction based on whichever one has a positive projection on the gradient descent direction, then update the heading as fast as possible to match our heading to the gradient descent heading. If we assume it is a kinematic system, the inputs are the linear and angular velocity.

We define the angle corresponding to the negative gradient of the navigation function θ_{ngi} as

$$\theta_{\text{ngi}} = \arctan\left(-\frac{\partial V_i^{X_i}}{\partial y_i}, -\frac{\partial V_i^{X_i}}{\partial x_i}\right), \tag{15}$$

and define the angle between the negative gradient and the heading to be $\theta_{\text{ni}} = \min\{\measuredangle(\theta_{\text{ngi}}, \theta_i), \measuredangle(\theta_{\text{ngi}}, -\theta_i)\}$. θ_{ni} is in the set $[-\pi/2, \pi/2]$ by construction (please refer to Figure 5). The angular velocity control law is given by

$$\dot{\theta}_i = \beta \frac{\theta_{\text{ngi}} - \theta_i}{\left|\theta_{\text{ngi}} - \theta_i\right|}, \tag{16}$$

and the linear velocity control law is given by

$$\dot{q}_i = -\alpha \frac{\left(\partial V_i^{X_i}/\partial q_i\right) B_{\text{ni}}^T(q)}{\left|\partial V_i^{X_i}/\partial q_i\right|}, \tag{17}$$

where α and β are constant speed parameters and

$$B_{\text{ni}} = \begin{bmatrix} \cos\theta_{\text{ni}} & -\sin\theta_{\text{ni}} \\ \sin\theta_{\text{ni}} & \cos\theta_{\text{ni}} \end{bmatrix}. \tag{18}$$

Figure 6(a) shows the effectiveness of the algorithm in a multiple obstacle avoidance task. A nonholonomic robot successfully reaches the target after transversing a workspace which is packed with randomly generated obstacles of different size. Figure 6(b) is a close-up view of Figure 6(a) showing fairly smooth motion of the robot around the obstacles.

7. Both Nonholonomic Constraints and Moving Obstacles

In this section, we consider the motion control problem subject to both nonholonomic constraints and moving obstacles. We will first find a feasible direction set such that all directions in this set can move the robot towards the goal and at same time away from the moving obstacles. We define the optimal feasible direction as the direction that is in the feasible set and closest to the gradient descent direction. If the robot's forward direction is in the feasible set, the robot will move forward; if the robot's backward direction is in the feasible set, the robot will move backward (note that backward direction and forward direction cannot be in the feasible set at the same time). When neither of these directions is in the feasible set, the robot will not generate linear velocity but will rotate towards the optimal feasible direction. In this way, the robot can move towards (taking both positions and orientations into account) the goal and avoid moving obstacle at same time.

If we use Θ_{oi} to denote the optimal rotation angle, the following formula for Θ_{oi} is given by

$$\Theta_{\text{oi}} = \begin{cases} 0 & \text{if } \Theta_{\min} \le 0,\ \Theta_{\max} \ge 0, \\ \Theta_{\min} & \text{if } \Theta_{\min} > 0, \\ \Theta_{\max} & \text{if } \Theta_{\max} < 0, \end{cases} \tag{19}$$

(refer to Figure 13) and we define the feasible steepest descent direction d_i to be

$$d_i = d_{\text{ngi}} B_{\text{oi}}^T(q), \tag{20}$$

where $B_{\text{oi}}(q)$ is defined as follows:

$$B_{\text{oi}}(q) = \begin{bmatrix} \cos\Theta_{\text{oi}} & -\sin\Theta_{\text{oi}} \\ \sin\Theta_{\text{oi}} & \cos\Theta_{\text{oi}} \end{bmatrix}. \tag{21}$$

Motion Control Law. The (mode-dependent) kinematics of each agent is then given by

$$\dot{\theta}_i = \beta \frac{\theta_{\text{roi}}}{|\theta_{\text{roi}}|}, \tag{22}$$

where θ_{roi} is defined as the angle from the orientation θ_i to the optimal feasible direction d_i:

$$\dot{q}_i = -\alpha \frac{\left(\partial V_i^{X_i}/\partial q_i\right) B_{\text{ai}}^T(q)}{\left|\partial V_i^{X_i}/\partial q_i\right|}, \tag{23}$$

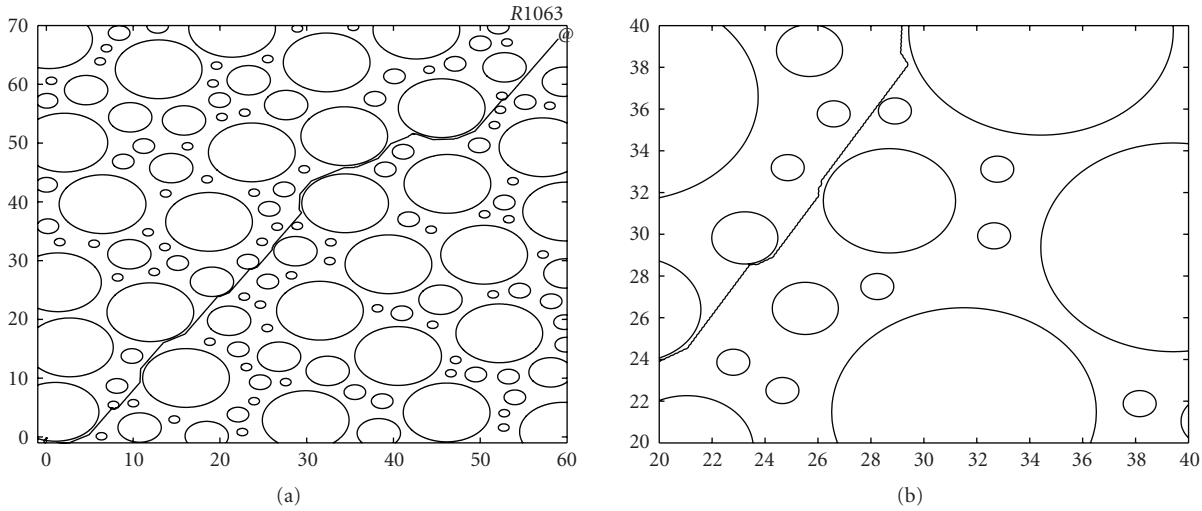


Figure 6: Motion planning for nonholonomic robots with multiple obstacles.

where the rotation matrix $B_{\text{ai}}(q_i)$ is defined as

$$B_{\text{ai}}(q) = \begin{bmatrix} \cos\Theta_{\text{ai}} & -\sin\Theta_{\text{ai}} \\ \sin\Theta_{\text{ai}} & \cos\Theta_{\text{ai}} \end{bmatrix}, \tag{24}$$

where Θ_{ai} is defined as the rotation angle from the negative gradient direction to the *actual* motion direction (forward direction or backward direction). Note that we require Θ_{ai} to be in the set $[-\pi/2, \pi/2]$; the robot will move forward if forward direction is a feasible direction, move backward if backward direction is a feasible direction. B_{ai} is set to a zero matrix when neither the forward direction nor the backward direction is in the feasible set. In this case, the robot will not have any linear velocity.

Figure 7 shows the comparison of two scenarios,the first scenario is the goal reaching task with no obstacles. The second scenario is the goal reaching task with one moving obstacle that is always in the way of the robot at each step of its motion. From the figures, it is clear that the robot can reach the goal in six steps if the route is clear, while needs 80 steps under the influence of the moving obstacle.

Figure 8 shows the difference of robot motions with and without moving obstacles in the task of static obstacle avoidance. The first figure shows that the robot can get around the static obstacle in 60 steps, however with a moving obstacle, the robot in the second figure needs 90 steps.

8. Simulation Results

In this section, we show how this framework succeeds in a typical navigation and coordination task: search-and-transport. We have performed simulations of agent teams in several environments. Figure 9 shows the results of a typical simulation, played in a 10×10 grid containing 6 static obstacles (represented with the # symbol) and 2 moving obstacles (represented with the *M* symbol) with a team of 3 agents trying to find 3 small objects (represented by the @ symbol) and 3 big objects (represented by the *O* symbol). The *O* symbols also represent positions where the big objects are picked up, and the @ symbols also represent positions where the small objects are picked up. The solid lines near the moving obstacles are trajectories of the moving obstacles. The robot $R1$ starts at position $(1, 0)$, the robot $R2$ starts at position $(3, 0)$, and the robot $R3$ starts at position $(5, 0)$. The small object collection location is $(9, 0)$ and the big object collection location is $(1, 0)$.

In Figure 9, we illustrate the search-and-transport task. Only $R3$'s trajectory is shown (represented by dashed line) in Figure 9 to reduce clutter.

Figures 10(a)–10(c) is part of the simulation. We use it to illustrate coordination among agents by showing the paths followed by the three agents and moving obstacles. The trajectories of agents $R1$, $R2$, and $R3$ are represented by the solid line, dash-dot line and dashed line, respectively. From this figure, we can see that $R2$ found a big object at the "*O*" point, $R3$ came to help and they returned the big object back to the big object collection position together. $R1$ found a small object at the "@" point, and is carrying it back to the small object collection position. The figure also illustrates how all agents avoid both static and moving obstacles at all times.

Figure 11 illustrates our moving obstacle avoidance technique. When $R1$ spots a moving obstacle represented by the M symbol, a moving obstacle avoidance algorithm is incorporated into the navigation function. By always moving along the feasible direction, $R1$ can make progress to the goal and avoid collision with the obstacle at the same time.

In Figure 12 we illustrate how our method works with a more complex scenario. The environment is enlarged from 10 by 10 to 50 by 50. The workspace is more clustered and the obstacles have a larger size and assume different shapes. It is shown that all the algorithms presented in the paper are not affected by these changes. The three robots can still coordinate and accomplish the search-and-transport task. The new workspace contains eight closely placed static obstacles of different shapes and two moving

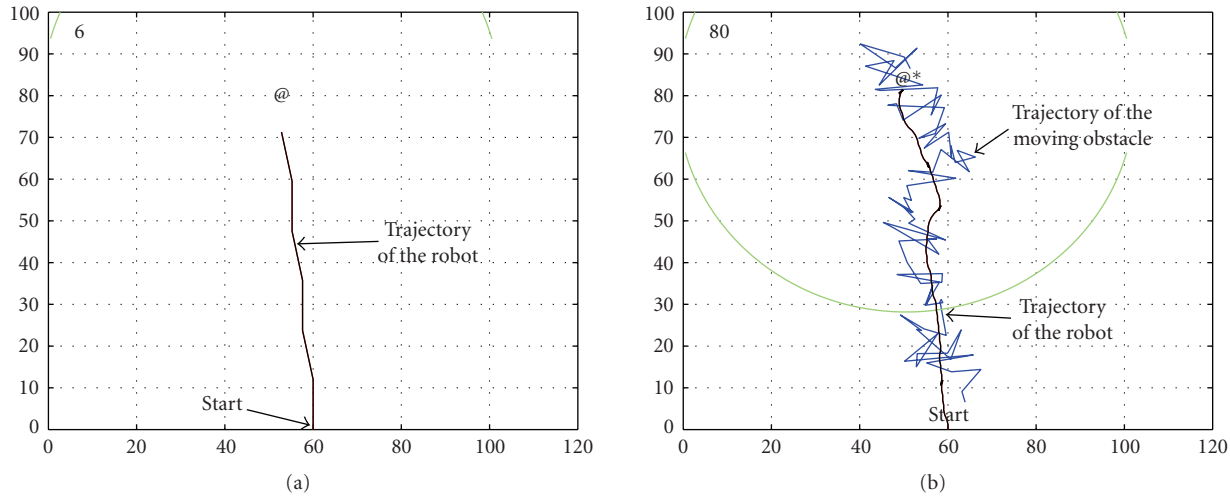


Figure 7: (a) shows the robot motion with nonholonomic constraints, no obstacle. (b) shows the robot motion with nonholonomic constraints, with a random moving obstacle, note that to fully test our algorithm, we keep the moving obstacle activated for each step.

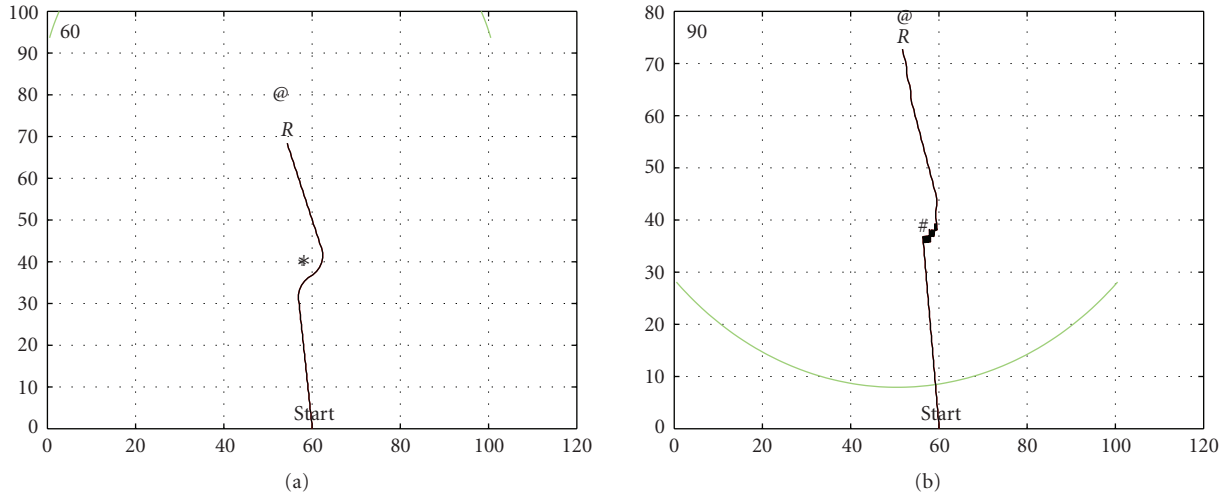


Figure 8: (a) shows the robot motion with nonholonomic constraints, with a static obstacle at (55, 40), (b) shows the robot motion with nonholonomic constraints, with a moving obstacle moving around (55, 40).

obstacles (represented with the M symbol). The number of large objects remains three and the number of small objects is increased to four. As in the last scenario, the solid lines near the moving obstacles are the trajectories of the moving obstacles. The robot $R1$ starts at position $(5, 0)$, the robot $R2$ starts at position $(15, 0)$, and the robot $R3$ starts at position $(25, 0)$. The small object collection location is $(45, 0)$ and the big object collection location is $(5, 0)$.

9. Team Stability Analysis

The definition of our modal navigation functions, and of our mode-switching rules, allows us to show that the system (entire agent team) is globally stable at all times and in all states [16].

With the mode switching technique defined in [6], we can guarantee the mode switches occur in finite time as follows. Although the order of mode switches cannot be predicted in advance, our switching technique guarantees that the mode-transition graph will be *acyclic* since mode switches only occur when the team has made progress towards solving the overall task and there is never a situation where a given mode is reentered. Therefore it is straightforward to show that our hybrid system is one of the switched system defined in [16].

Now we only need to show that the low-level path planning scheme presented here is stable.

9.1. Stability without Constraints. Within each mode, we can define a mode-specific Lyapunov function of the following

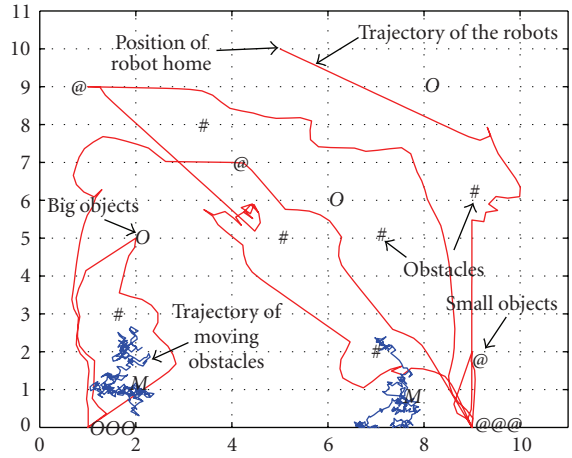


FIGURE 9: This figure is the snapshot after the task is completed. All three big objects are returned to the big object collection position and all three small objects are returned to the small object collection position. The robots are back in their home position. The dashed line represents the trajectory of agent $R3$.

form:

$$V^X(q) = \sum_{i=1}^{Q} V_i^{X_i}(q) - \sum_{i=1}^{Q}\sum_{j=i+1}^{Q} VR\left(q_i, q_j\right) \tag{25}$$

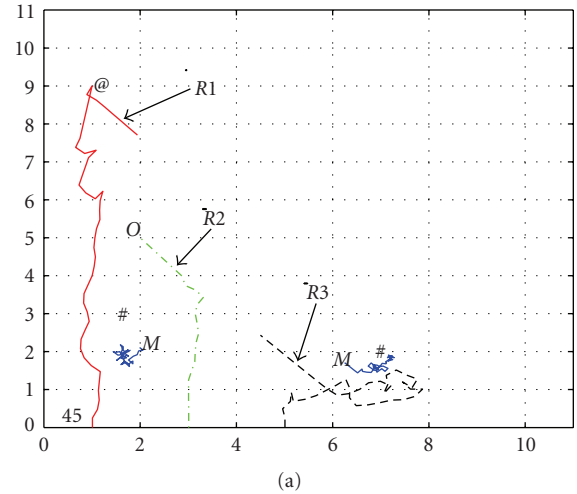
$$= \sum_{i=1}^{Q}\left(VA^{X_i}(q_i) + VO^{X_i}(q_i)\right) + \sum_{i=1}^{Q}\sum_{j=i+1}^{Q} VR\left(q_i, q_j\right), \tag{26}$$

where the step from (25) to (26) is justified by the fact that $VR(q_i, q_j) = VR(q_j, q_i)$. In these equations Q is the number of robots.

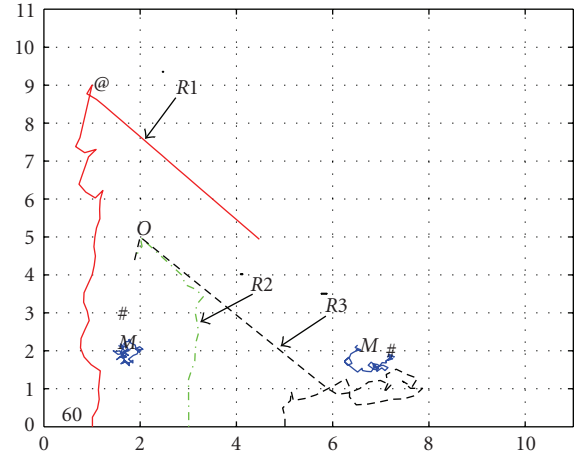
To show intramodal Lyapunov stability, we are required to show $V^X(q) \geq 0$ for all q and $\dot{V}^X(q) < 0$ for all q, t. $V^X(q) \geq 0$ follows naturally from the definition of $V^X(q)$ in (26). To show that $V^X(q)$ is always decreasing, we begin by using (25). For convenience, in the derivations that follow, we have replaced terms of the following form $VR(q_i, q_j)$ with the short form VR_{ij}:

$$\dot{V}^X(q) = \sum_{i=1}^{Q} \frac{\partial V_i^{X_i}(q)}{\partial q_i}\dot{q}_i^T + \sum_{i=1}^{Q}\sum_{i\neq j} \frac{\partial V_i^{X_i}(q)}{\partial q_j}\dot{q}_j^T - \sum_{i=1}^{Q}\sum_{j=i+1}^{Q}\left(\frac{\partial VR_{ij}}{\partial q_i}\dot{q}_i^T + \frac{\partial VR_{ij}}{\partial q_j}\dot{q}_j^T\right). \tag{27}$$

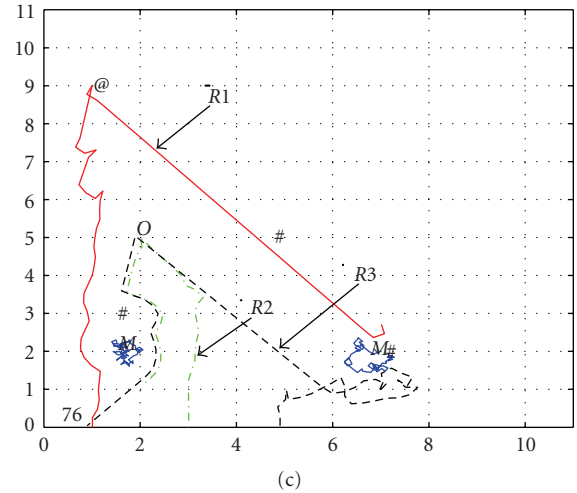
Now, observe that since q_j only appears in $V_i^{X_i}(q)$ through the VR_{ij} terms, the second term of (27) can be



(a)



(b)



(c)

FIGURE 10: This figure is to show the agent team coordination in a big object carrying task. In (a) $R2$ finds a big object, stops searching and calls for help. In (a) $R3$ comes to help after receiving the *confirm* message. In (b) $R3$ arrives at the big object and $R2$ and $R3$ are carrying the big object home together. Finally, in (c) $R2$ and $R3$ reached the object home.

rewritten as

$$\begin{aligned}\sum_{i=1}^{Q}\sum_{i\neq j}\frac{\partial V_i^{X_i}}{\partial q_j}\dot{q}_j^T &= \sum_{i=1}^{Q}\sum_{i\neq j}\frac{\partial VR_{ij}}{\partial q_j}\dot{q}_j^T \\ &= \sum_{i=1}^{Q}\sum_{j=i+1}^{Q}\frac{\partial VR_{ij}}{\partial q_j}\dot{q}_j^T + \sum_{i=1}^{Q}\sum_{j=1}^{i-1}\frac{\partial VR_{ij}}{\partial q_j}\dot{q}_j^T \\ &= \sum_{i=1}^{Q}\sum_{j=i+1}^{Q}\frac{\partial VR_{ij}}{\partial q_j}\dot{q}_j^T + \sum_{p=1}^{Q}\sum_{s=p+1}^{Q}\frac{\partial VR_{ps}}{\partial q_p}\dot{q}_p^T \\ &= \sum_{i=1}^{Q}\sum_{j=i+1}^{Q}\left(\frac{\partial VR_{ij}}{\partial q_i}\dot{q}_i^T + \frac{\partial VR_{ij}}{\partial q_j}\dot{q}_j^T\right).\end{aligned} \tag{28}$$

Thus, the second and third terms in (27) will cancel, and we will have

$$\dot{V}^X(q) = \sum_{i=1}^{Q}\frac{\partial V_i^{X_i}(q)}{\partial q_i}\dot{q}_i^T. \tag{29}$$

If we substitute for $\dot{q}_i$ using the dynamics defined in (7), we will have

$$\begin{aligned}\dot{V}^X(q) &= -\alpha\sum_{i=1}^{Q}\frac{\partial V_i^{X_i}(q)}{\partial q_i}\frac{\left(\partial V_i^{X_i}/\partial q_i\right)^T}{\left|\partial V_i^{X_i}/\partial q_i\right|} \\ &= -\alpha\sum_{i=1}^{Q}\left|\frac{\partial V_i^{X_i}(q)}{\partial q_i}\right|,\end{aligned} \tag{30}$$

and we will have $\dot{V}^X(q) < 0$ for all t, q as required.

Remark 1. The constructed Lyapunov function $V^X(q)$ is a generalized energy function. The first term $\sum_{i=1}^{Q} VA^{X_i}(q_i)$ decreasing means that the agents get closer to the attractors. The second term $\sum_{i=1}^{Q} VO^{X_i}(q_i)$ decreasing means that the agents move away from the static obstacles. The third term $\sum_{i=1}^{Q}\sum_{j=i+1}^{Q} VR(q_i, q_j)$ decreasing means that the agents try to stay away from the other team members.

9.2. Stability Subject to Nonholonomic Constraints. If we substitute into (29) for $\dot{q}_i$ using the dynamics defined in (17), we will have

$$\dot{V}^X(q) = -\sum_{i=1}^{Q}\left(\frac{\partial V_i^{X_i}(q)}{\partial q_i}\right)\alpha B_{\mathrm{ni}}(q)\frac{\left(\partial V_i^{X_i}/\partial q_i\right)^T}{\left|\partial V_i^{X_i}/\partial q_i\right|}. \tag{31}$$

If we denote $v_{xi} = -\partial V_i^{X_i}(q)/\partial x_i$ and $v_{yi} = -\partial V_i^{X_i}(q)/\partial y_i$,

$$\begin{aligned}&\left(\frac{\partial V_i^{X_i}(q)}{\partial q_i}\right)B_{\mathrm{ni}}(q)\left(\frac{\partial V_i^{X_i}}{\partial q_i}\right)^T \\ &\quad= \begin{bmatrix}v_{xi}\\ v_{yi}\end{bmatrix}^T B_{\mathrm{ni}}\begin{bmatrix}v_{xi}\\ v_{yi}\end{bmatrix} \\ &\quad= \begin{bmatrix}v_{xi}\\ v_{yi}\end{bmatrix}^T\begin{bmatrix}\cos\theta_{\mathrm{ni}} & -\sin\theta_{\mathrm{ni}}\\ \sin\theta_{\mathrm{ni}} & \cos\theta_{\mathrm{ni}}\end{bmatrix}\begin{bmatrix}v_{xi}\\ v_{yi}\end{bmatrix} \\ &\quad= \left(v_{xi}^2\cos\theta_{\mathrm{ni}} + v_{yi}^2\cos\theta_{\mathrm{ni}}\right).\end{aligned} \tag{32}$$

Note that by construction, $\theta_{\mathrm{ni}} \subseteq [-\pi/2, \pi/2]$. Therefore we will have $\dot{V}^X(q) < 0$ for all t, q as required.

9.3. With Moving Obstacles. From (29), we have

$$\dot{V}^X(q) = \sum_{i=1}^{Q}\frac{\partial V_i^{X_i}(q)}{\partial q_i}\dot{q}_i^T. \tag{33}$$

If we substitute for $\dot{q}_i$ using the dynamics defined in (14), we will have

$$\dot{V}^X(q) = \sum_{i=1}^{Q}\frac{\partial V_i^{X_i}(q)}{\partial q_i}kd_{\mathrm{fngi}}. \tag{34}$$

According to the definition of feasible directions (refer to (9)), we have $\dot{V}^X(q) < 0$ for all t, q as required.

9.4. With Moving Obstacles and Nonholonomic Constraints. If we substitute for $\dot{q}_i$ using the dynamics defined in (23), we will have

$$\dot{V}^X(q) = (-\alpha)\sum_{i=1}^{Q^X}\left(\frac{\partial V_i^{X_i}(q)}{\partial q_i}\right)B_{\mathrm{ai}}(q)\frac{\left(\partial V_i^{X_i}/\partial q_i\right)^T}{\left|\partial V_i^{X_i}/\partial q_i\right|}. \tag{35}$$

If we denote $v_{xi} = -\partial V_i^{X_i}(q)/\partial x_i$ and $v_{yi} = -\partial V_i^{X_i}(q)/\partial y_i$,

$$\begin{aligned}&\left(\frac{\partial V_i^{X_i}(q)}{\partial q_i}\right)B_{\mathrm{ai}}(q)\left(\frac{\partial V_i^{X_i}}{\partial q_i}\right)^T \\ &\quad= \begin{bmatrix}v_{xi}\\ v_{yi}\end{bmatrix}^T B_{\mathrm{ai}}(q)\begin{bmatrix}v_{xi}\\ v_{yi}\end{bmatrix} \\ &\quad= \begin{bmatrix}v_{xi}\\ v_{yi}\end{bmatrix}^T\begin{bmatrix}\cos\theta_{\mathrm{ai}} & -\sin\theta_{\mathrm{ai}}\\ \sin\theta_{\mathrm{ai}} & \cos\theta_{\mathrm{ai}}\end{bmatrix}\begin{bmatrix}v_{xi}\\ v_{yi}\end{bmatrix} \\ &\quad= \left(v_{xi}^2\cos\theta_{\mathrm{ai}} + v_{yi}^2\cos\theta_{\mathrm{ai}}\right).\end{aligned} \tag{36}$$

Note that by construction, $\theta_{\mathrm{ai}} \subseteq [-\pi/2, \pi/2]$. Therefore we will have $\dot{V}^X(q) < 0$ for all t, q as required.

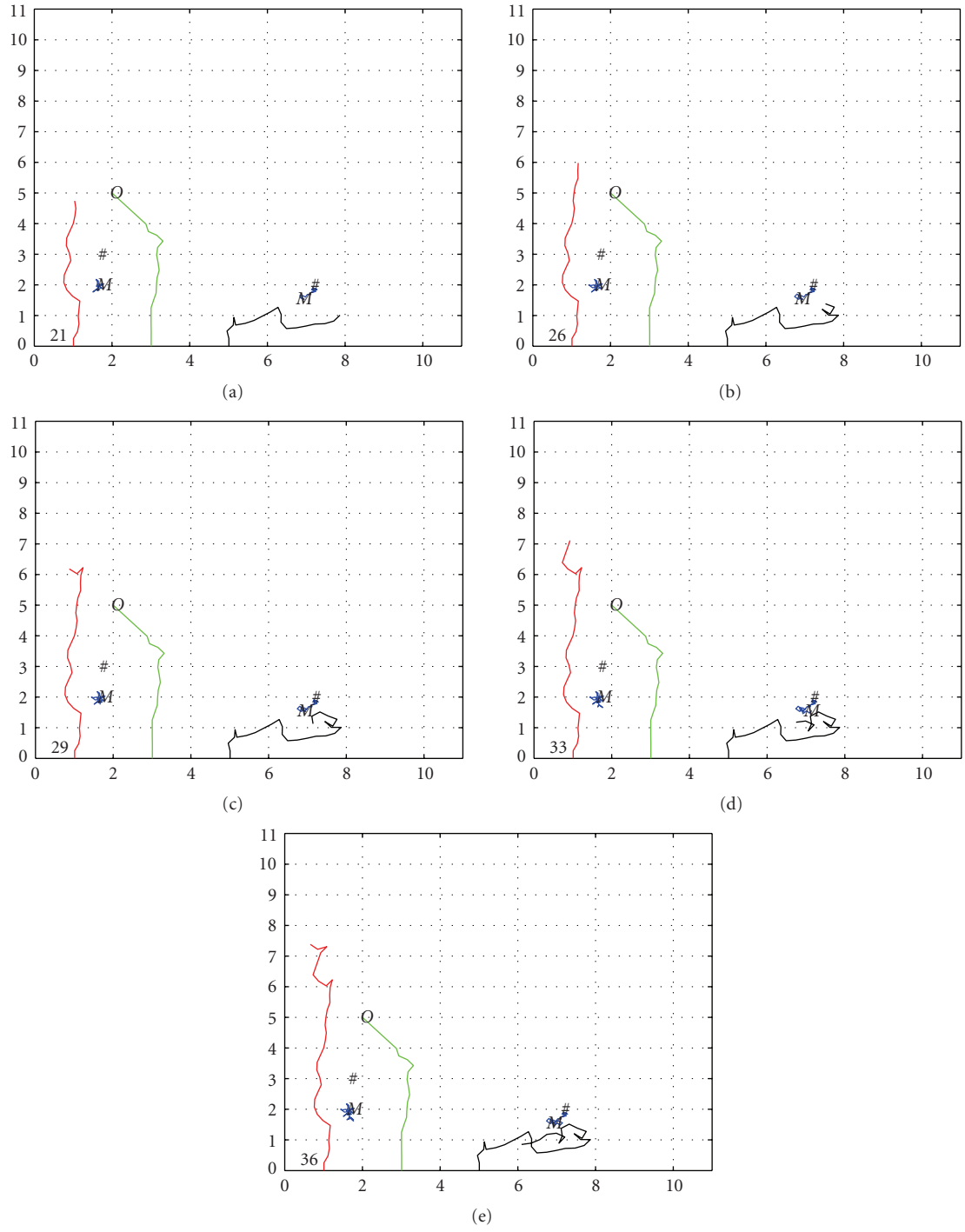


FIGURE 11: (a) $R1$ spots a moving obstacle in the way. and start moving obstacle avoidance algorithm for their navigation. (b)–(e) shows for each step, the robot is moving away from the current position the obstacle therefore avoiding collision. Meanwhile the total effects of attractive "force" from the unsearched cells and the objects attract the robot towards to the goal.

10. Conclusions and Future Work

In our previous work, we proposed a software framework and a hybrid navigation scheme for multiple agent navigation and coordination. However, there were several issues that remained unresolved in the previous work. Specifically, the robots and their environment were abstractions of the real world. For instance, the robots had the ability to move in all directions and the environment contained only fixed obstacles. In this paper, we extend our work by eliminating these ideal characteristics and considering the challenges that robots are likely to encounter in the real world. In

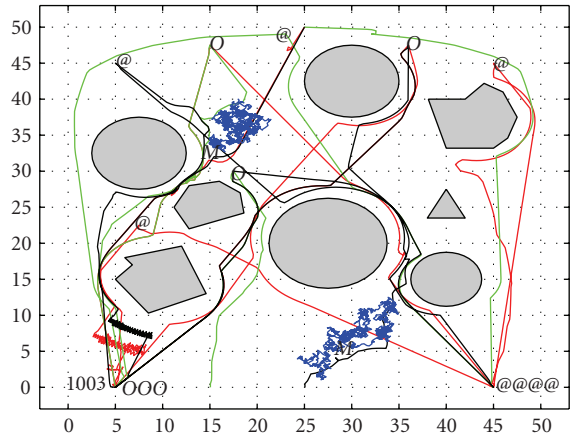


FIGURE 12: We show the simulation results with a more complex environment. Obstacles are larger and take on different shapes, and the workspace is more crowded. This figure is the snapshot after the task is completed. All three big objects are returned to the big object collection position and all four small objects are returned to the small object collection position.

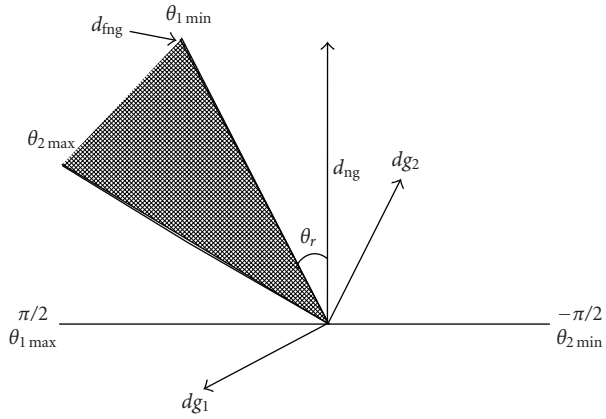


FIGURE 13: This figure illustrates our optimal feasible direction scheme. Note that the filled area is the resulting feasible area.

particular, we propose geometric-based methods to avoid moving obstacles and satisfy nonholonomic constraints. Our methods are suitable for fast online calculation and they are easily combined with one another to solve more complex problems such as nonholonomic constraints with moving obstacles. Furthermore, our control framework applies to a wide range of scenarios. We have demonstrated the search-and-transport problems with interdependencies for a multiagent team. And this framework is applicable to a much wider range of tasks than were discussed in this paper, due to our hierarchical design and fully distributed organization and because our stability proof makes no assumptions about the nature of the team members.

Appendix

Proof. (For simplicity, we drop the subscript "i" which refers to the ith robot in this proof.)

First we denote the angle corresponding to the gradient of constraint g_j by θ_{vj} and define it as follows:

$$\theta_{vj} = \arctan\left(\frac{\partial g_j}{\partial y}, \frac{\partial g_j}{\partial x}\right), \tag{A.1}$$

where $-\pi < \theta_{vj} \leq \pi$. To better illustrate our method, we represent all directions in the following form of rotation angles with respect to d_{ng}. Θ_j is defined as the angle between d_{ng} and dg_j. We define this angle to be positive if a counterclockwise rotation takes d_{ng} to dg_j and negative otherwise:

$$\Theta'_j = \theta_{vj} - \theta_{\text{ng}}. \tag{A.2}$$

The resultant Θ'_j falls in the interval $[-2\pi, 2\pi]$ and we transform it to the interval $[-\pi, \pi]$ by the following formula:

$$\Theta_j = \begin{cases} \Theta'_j & \text{if } -\pi \leq \Theta'_j \leq \pi, \\ \Theta'_j - 2\pi & \text{if } \Theta'_j > \pi, \\ \Theta'_j + 2\pi & \text{if } \Theta'_j < -\pi. \end{cases} \tag{A.3}$$

Next we construct a feasible direction set $[\Theta_{j\,\min}, \Theta_{j\,\max}]$ for each of the activated moving obstacles:

$$\Theta_{j\,\min} = \begin{cases} \Theta_j - \dfrac{\pi}{2} & \text{if } \Theta_{\text{j}} \geq 0, \\ -\dfrac{\pi}{2} & \text{if } \Theta_{\text{j}} < 0, \end{cases}$$
$$\Theta_{j\,\max} = \begin{cases} \Theta_j + \dfrac{\pi}{2} & \text{if } \Theta_{\text{j}} \leq 0, \\ \dfrac{\pi}{2} & \text{if } \Theta_{\text{j}} > 0. \end{cases} \tag{A.4}$$

Note that the feasible direction set $[\Theta_{j\,\min}, \Theta_{j\,\max}]$ for each activated moving obstacle constraint is a subset of $[-\pi/2, \pi/2]$ because all feasible directions must lie in the same half-tangent plane as d_{ng}. Next, we find the intersection $\theta_f = [\Theta_{\min}, \Theta_{\max}]$ of the feasible direction sets $[\Theta_{j\,\min}, \Theta_{j\,\max}]$ that works for all moving obstacles:

$$\Theta_{\min} = \max\{\Theta_{1\,\min}, \Theta_{2\,\min}, \ldots, \Theta_{m\,\min}\},$$
$$\Theta_{\max} = \min\{\Theta_{1\,\max}, \Theta_{2\,\max}, \ldots, \Theta_{m\,\max}\}. \tag{A.5}$$

By construction, we know that θ_f is also a subset of $[-\pi/2, \pi/2]$. □

References

[1] S. Hirose, S. Yokota, A. Torii, et al., "Quadruped walking robot centered demining system—development of TITAN-IX and its operation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '05)*, vol. 2005, pp. 1284–1290, 2005.

[2] L. E. Parker, "ALLIANCE: an architecture for fault tolerant multirobot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[3] A. Davids, "Urban search and rescue robots: from tragedy to technology," *IEEE Transactions on Intelligent Systems*, vol. 17, no. 2, pp. 81–83, 2002.

[4] B. P. Gerkey and M. J. Mataric, "Sold!: auction methods for multirobot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 758–768, 2002.

[5] A. J. Ijspeert, A. Martinoli, A. Billard, and L. M. Gambardella, "Collaboration through the exploitation of local interactions in autonomous collective robotics: the stick pulling experiment," *Autonomous Robots*, vol. 11, no. 2, pp. 149–171, 2001.

[6] J. Ren, K. A. McIsaac, and R. V. Patel, "A novel hybrid navigation scheme for reconfigurable multi-agent teams," *International Journal of Robotics and Automation*, vol. 21, no. 2, pp. 100–109, 2006.

[7] H. Ghenniwa and M. Kamel, "Interaction devices for coordinating cooperative distributed systems," *Intelligent Automation and Soft Computing*, vol. 6, no. 3, pp. 173–184, 2000.

[8] J.-O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 338–349, 1992.

[9] R. Kimmel, N. Kiryati, and A. M. Bruckstein, "Multivalued distance maps for motion planning on surfaces with moving obstacles," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 3, pp. 427–436, 1998.

[10] K. Fujimura and H. Samet, "Hierarchical strategy for path planning among moving obstacles," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 1, pp. 61–69, 1989.

[11] R. A. Conn and M. Kam, "Robot motion planning on N-dimensional star worlds among moving obstacles," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 320–325, 1998.

[12] J. M. Esposito and V. Kumar, "A method for modifying closed-loop motion plans to satisfy unpredictable dynamic constraints at run-time," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02)*, pp. 1691–1696, Washington, DC, USA, 2002.

[13] M. Bazaraa, H. Sherali, and C. Shetty, *Nonlinear Programming Theory and Algorithms*, John Wiley & Sons, New York, NY, USA, 1995.

[14] H.-S. Shim and Y.-G. Sung, "Stability and four-posture control for nonholonomic mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 148–154, 2004.

[15] J. Evans, B. Krishnamurthy, B. Barrows, T. Skewis, and V. Lumelsky, "Handling real-world motion planning: a hospital transport robot," *IEEE Control Systems Magazine*, vol. 12, no. 1, pp. 15–19, 1992.

[16] M. S. Branicky, "Multiple Lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 475–482, 1998.