Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
November-December 2016: 1–12

DOI: 10.1177/1729881416663666
arx.sagepub.com
SAGE

Multi-robot patrol: A distributed algorithm based on expected idleness

Chuanbo Yan and Tao Zhang

Abstract

Multi-robot with advantages of spatial distribution and fault tolerance is competent for patrol missions and has the potential to be used in security and surveillance applications. This article focuses on the frequency-based patrol designed to guarantee the frequent access to key positions in the environment. A distributed algorithm based on expected idleness is proposed, aiming to promote the efficiency of cooperation, which remains to be fault tolerant and scalable. The expected idleness is estimated with information shared between robots and utilized to avoid conflicts in the decision process. Comparisons with state-of-the-art algorithms have been conducted in a realistic simulator, Stage; moreover, the fault tolerance and scalability have also been tested. Experiments on real robots have further verified the applicability of the proposed algorithm.

Keywords

Multi-robot patrol, patrolling, distributed algorithms, fault tolerance, scalability

Date received: 23 April 2016; accepted: 17 June 2016

Topic: Special Issue - Manipulators and Mobile Robots
Topic Editor: Michal Kelemen

Introduction

Patrol is the monitoring task needed to go around or through a specified area frequently, which can be employed in environmental monitoring, information collection, intrusion detection and other security or surveillance applications. But sometimes, the patrol mission may be tedious or even dangerous, so as the advancement in robotics research in recent decades, it is expected that mobile robots can be a substitute for humans in such mission.[1,2] Compared with a single robot, multi-robot system is more competent for patrolling, due to the spatial distribution, scalability and fault tolerance brought by redundancy.

Research on multi-robot patrol can be grouped into two categories: adversarial patrol and frequency-based patrol. The adversarial patrol[3] aims to detect the possible intruders who attempt to access the protected area, and the patrol path is a circular boundary in most cases. On the other hand, the frequency-based patrol[4] is designed to guarantee the frequent access to the specified area in order to gather information in time. Normally, the patrol area can be abstracted into a topological map, equivalent to an undirected graph in graph theory. Robots are capable of sensing within a certain range, so they can just visit some key positions (equivalent to the vertices of the graph) instead of covering the entire area. The performance is usually measured by the time interval between two visits to the same position, which is referred to as idleness. This article mainly concentrates on the latter problem, and the expression multi-robot patrol refers to frequency-based patrol without any instructions.

From the optimization theory point of view, the multi-robot patrol problem has a close connection with the vehicle routing problem (VRP)[5] and multiple traveling salesman problem.[6] However, these combinatorial

Department of Automation, Tsinghua University, Beijing, China

Corresponding author:
Tao Zhang, Tsinghua University, FIT1-110, Beijing, 100084, China.
Email: taozhang@mail.tsinghua.edu.cn

optimization problems do not consider the repetitive and dynamic characteristics of the patrol problem, nor the cooperation between robots during patrolling.[7] The algorithms of multi-robot patrol can be divided into two categories: centralized algorithms and distributed algorithms. Two typical centralized algorithms[8,9] are cyclic patrolling based on the cycle in the graph and independent patrolling based on graph partition. The centralized algorithms are able to achieve good enough solutions in most cases. However, the cycle is not easy to find out especially in sparse topology graphs, and the partition may be unbalanced under certain configurations; moreover, the offline planning procedure of centralized algorithms is unable to adapt to possible changes in time. In distributed algorithms,[10] each robot independently chooses patrol route during the online planning procedure using greedy strategies or task allocation methods with limited communications. Compared with centralized algorithms, although distributed algorithms often result in worse performance, but they can offer better fault tolerance and scalability. More specifically, they can adapt to environments with diverse topological structure, and can also tolerate robots to join or withdraw during patrolling.

This article proposes a distributed algorithm based on expected idleness, aiming at the effective implementation of multi-robot patrol. The expected idleness is estimated with shared information through distributed communication and utilized to avoid potential conflicts between robots in the decision process. In this way, the simple and reactive algorithm in this article can provide a competent solution for patrolling compared with the previous studies. The algorithm is executed on each robot independently without the need for offline planning or centralized control, thus offers fault tolerance and scalability for the robot team.

The rest of this article is organized as follows. The literature review and problem formulation are presented in sections 'Related work' and 'Problem formulation'. Then, the proposed algorithm is described in section 'Distributed patrol algorithm based on expected idleness'. Sections 'Experiments' and 'Simulation tests about scalability and fault tolerance' report the results of simulation and real robot experiments. Finally, this article ends with conclusions in section 'Conclusions'.

Related work

Research on multi-robot patrol issue uses the model of undirected graph to represent the patrol area. The vertices of the graph correspond to the key positions needed to be visited frequently, and the edges and their weight stand for connections and path lengths between these positions. Thus, algorithms from the graph theory can be utilized to solve the patrol problem with appropriate modifications, and this is the basic idea of most centralized algorithms. Centralized algorithms relied on graph methods can be divided into two categories: the cyclic algorithms and the partition-based algorithms.

In cyclic algorithms, robots move along a precalculated cyclic path through all vertices in the graph. The cyclic path can be calculated as a traveling salesman problem (TSP) problem (NP-hard problem can use heuristic algorithms to reach near-optimal solutions). Chevaleyre[8] proved that there exists an upper boundary of maximum idleness for the cyclic path based on the TSP solution. He also pointed out that the partition-based algorithms are preferred when graphs have long edges separating several regions. Further analysis and comparative study between the cyclic algorithms and the partition-based algorithms are provided in the study by Portugal et al.[9] Elmaliach et al.[4] assigned robots evenly in travel time along the cyclic path, so as to guarantee all the vertices to be visited under uniform frequency. For maps with appropriate cyclic path, homogeneous robots coordinated tightly to maintain uniform temporal separation can be efficient, but such method for heterogeneity of robots (with different velocities) under unreliable communication performs poorly.[11]

The partition-based algorithms make a partition of the map into several disjoint regions and assign each robot a region to patrol independently. Furthermore, the graph partitioning and the local route planning could be optimized together by minimizing the local route with the maximum length.[9] In this way, the partition-based patrolling could be construed as a variant of the VRP.[5] Multilevel subgraph patrolling algorithm[12] is a typical partition-based algorithm, which uses a multilevel method for partitioning graphs,[13] and optimizes the local patrolling route of each robot. The performance of this algorithm mainly relies on whether the graph partition is balanced enough.[14] In partition-based algorithms, each robot patrols inside its own region after the partition of the graph is completed during the initialization phase. In the course of patrolling, some robots may not perform as expected or even be broken, and original partition-based algorithms could not cope with such situations. Pippin et al.[15] used an external monitor to inspect the visits of each vertex and adjust the task assignment among robots through auction in order to achieve better balance of robots' payload. This method introduces the process of online decision-making to improve the robustness, which is the main idea of distributed patrol algorithms.

Different from the previous centralized algorithms, Pasqualetti et al.[7] analysed the patrol problem according to three categories of graph structure. In the case of a chain graph, they proposed a centralized polynomial time algorithm to compute the optimal trajectory and designed a distributed procedure to approach the optimal trajectory. In the case of a tree or cyclic graph, they focused on optimization problems rather than distributed algorithms. They proposed a polynomial time algorithm for the optimal trajectory of a tree graph. They showed that the optimization problem for the graph with cyclic structure is NP-hard and proposed approximate algorithms accordingly. The issue of patrolling a set of

vertices with different priorities is considered in the study by Pasqualetti et al.[16] A closed path through all vertices is constructed with graph-theoretic methods, and a Stop-Go team trajectory, which optimizes the weighted idleness, is presented for robots along the path.

Incipient research of distributed patrol algorithms uses intuitive methods to guide individual robots towards left positions unvisited for some time, which are discussed and compared in the studies by Portugal and Rocha,[14] Machado et al.[17] and Almeida et al.[18] Task allocation, auction mechanism and game theory have also been used for distributed patrolling.[19–21] In distributed algorithms, each robot is offered with proper autonomy without the usage of centralized planning, and the patrol routes are determined online to accommodate the possible changes over time. For example, the conscientious reactive (CR) algorithm chooses the next vertex to visit from neighbors of the current position, considering only the instantaneous idleness. Iocchi et al.[22] examined the performance of multi-robot patrol algorithms in a realistic simulator, Stage, investigated the impact of some realistic circumstances (e.g. conflicts or deadlocks between robots in narrow environments, robots with different velocities in coordinated cyclic patrolling) and demonstrated the necessity of online coordination for robust patrolling.

Portugal and Rocha[10] presented two distributed multi-robot patrol algorithms using Bayesian-based mathematical formalism, named greedy Bayesian strategy (GBS) and state exchange Bayesian strategy (SEBS). The GBS drives each robot to maximize the local benefit (the instantaneous idleness divided by estimated time of movement) without negotiation with other robots. Furthermore, the SEBS enables the robot to consider the impact of other robots in the decision process, by sharing intention with robots around. The utility function is equivalent to the local benefit multiplied by a factor of intention. Both realistic simulations and real robot experiments are presented in their work. Typically, the SEBS performs better than the GBS and preceding distributed algorithms, due to the decrease of conflicts between robots. But the performance of SEBS would be affected when the number of robots changes dynamically, as some parameters in the algorithm need to be determined based on the number of robots.

Generally speaking, the distributed algorithms are less efficient than the centralized algorithms, mainly because of conflicts between robots. Further research to promote the efficiency of distributed patrol is still needed. The algorithm SEBS[10] takes into account the intentions of robots in order to prevent robots from competing for the same region. The work presented in this article exploits similar ideas but reduces the dependence on specific parameters (e.g. the number of robots and the upper bound of idleness in the SEBS algorithm). Comparisons with the state-of-the-art algorithms using Stage simulator and experiments on real robots show that the proposed algorithm is efficient in patrolling. The fault tolerance and scalability are also tested under various conditions through simulations using Stage.

Problem formulation

As mentioned earlier, the patrol area is represented as an undirected graph $G(V, E)$, in which $V = \{v_1 \ldots v_n\}$ stands for the set of vertices ($n = |V|$ is the cardinality of the set V) and $E = \{e_{i,j}\}$ stands for the set of edges. If there exists an edge $e_{i,j} \in E$ between vertex $v_i, v_j \in V$, the weight $c_{i,j}$ of the edge $e_{i,j}$ corresponds to the path length between this pair of vertices.

The graph $G(V, E)$ can be extracted from the occupancy grid map or the geometrical map. A common method is to generate the Voronoi diagram to represent the skeleton of the map[23] and to form vertices at the junction of edges.[24] In this article, the graph is assumed to be obtained in advance and known by all patrol robots.

To evaluate the performance of different patrol algorithms, idleness,[10] also referred to as refresh time,[7] which represents the time interval between two consecutive visits to the same vertex, is introduced as a criterion.

The idleness of a vertex $v_i \in V$ is calculated whenever the vertex is visited by a robot. Suppose vertex v_i is visited by a robot at time t_k for the k-th time, the idleness is calculated as

$$I_{v_i}(t_k) = t_k - t_{k-1}$$

where t_{k-1} represents the previous time instant that the vertex was visited.

Consequently, the average idleness of vertex v_i at time t_k is updated as

$$I_{v_i}^{avg}(t_k) = \frac{(k-1) \cdot I_{v_i}^{avg}(t_{k-1}) + I_{v_i}(t_k)}{k}$$

where k is the number of visits to vertex v_i until time t_k. Obviously, this is a recursive expression and can be calculated step by step.

The average idleness of the graph $G(V, E)$ can be calculated as

$$I_G^{avg}(t_k) = \frac{\sum_{i=1}^{n} I_{v_i}^{avg}(t_k)}{n} \tag{1}$$

The average idleness of the graph is an important criterion to evaluate the performance of different patrol algorithms. Another important criterion is the maximum average idleness of all vertices in the graph, which is defined as

$$\max\{I_V^{avg}(t_k)\} = \max_{v_i \in V = \{v_1 \ldots v_n\}} \{I_{v_i}^{avg}(t_k)\} \tag{2}$$

The patrol route of a robot can be expressed as an array of vertices to be visited sequentially. For a team of m robots, the set of patrol routes can be formed as

$$\mathbf{x} = \{x_1, \ldots, x_r, \ldots, x_m\}$$

where x_r is the patrol route of the r-th robot and can be expanded as

$$x_r = [v_{r,1}, v_{r,2}, \ldots], \quad v_{r,1}, v_{r,2}, \ldots \in V$$

The patrol mission requires the set of patrol routes $\mathbf{x}$ to coverage all the vertices in graph $G(V, E)$. So subject to constraint that

$$\forall v_i \in V, \quad \exists x_r \in \mathbf{x} : v_i \in x_r$$

The global objective function is to minimize the average idleness of the graph or the maximum average idleness of all vertices

$$f = \arg\min_x \{I_G^{avg} \text{ or } \max\{I_V^{avg}\}\} \tag{3}$$

In centralized algorithms, the patrol routes of all the robots are precalculated based on the static information about the environment and robots. In contrast, distributed algorithms enable each robot to choose patrol route dynamically and independently, according to the estimation of instantaneous information. The instantaneous idleness is the most commonly used information for online planning in distributed algorithms.

The instantaneous idleness $I_{v_i}^{ins}$ of a vertex v_i at time t is defined as the time elapsed since the last visit by a robot

$$I_{v_i}^{ins} = t - t_{last} \tag{4}$$

where t_{last} is the last time instant that the vertex was visited.

The last visit time t_{last} of every vertex in the graph $G(V, E)$ is shared among robots using distributed communication mechanism. When a robot reaches a vertex, it publishes a message of arrival to inform other robots to update their records. In the process of determining the next vertex to visit, the instantaneous idleness can be obtained from current time minus last visit time.

Note that the published messages may be delayed or lost and be limited by the range of communication. So the information maintained by a robot may be inaccurate estimation of the instantaneous idleness, and robust distributed algorithms need to tolerate such deviation.

Distributed patrol algorithm based on expected idleness

The proposed algorithm uses a similar decision process like other distributed patrol algorithms. After the current vertex is reached, the robot chooses the next vertex to visit from neighbors of the current vertex according to specific criteria. The algorithm is reactive rather than cognitive, because the robot only determines the next vertex instead of a path of several vertices, in order to cope with unforeseen situations while patrolling. Then, the robot publishes its next target vertex to notify other robots with the purpose of reducing conflicts (e.g. more than one robots move to the same vertex).

In the previous research, robots only share their next target vertex to avoid conflicts without considering their expected time of arrival. But conflicts occur mainly when robots approach the same vertex within a short period of time. In this article, the effect of time constraint is considered for the cooperation between robots. A reactive algorithm concerned with expected time is proposed and named as expected reactive (ER) algorithm .

In the proposed algorithm, the expected time of arrival is estimated by each robot and shared with other robots. Then, the expected idleness of vertex is calculated on the basis of collected information and used in the decision process. Meanwhile, the estimation of travel time is utilized to replace path length as the cost function, so as to adapt to different velocities of robots.

ER algorithm

Suppose the r-th robot reaches vertex $v_{r,k}$ at time t. The robot publishes the message of arrival $(v_{r,k}, t)$ to inform other robots to update their records of last visit time t_{last}. Then, the robot needs to decide the target vertex to visit next.

Each robot maintains the set S_{ints} that contains the target vertices of other robots and corresponding expected time of arrival. At the beginning of the decision process, the set S_{ints} is updated to filter out outdated intention information, which may cause by faults of communication or robot.

The robot computes the utilities of visiting all the adjacent vertices and chooses the maximum as the next target vertex. For an adjacent vertex $v_i \in N(v_{r,k})$, the travel time Δt between vertex $v_{r,k}$ and v_i is estimated as

$$\Delta t = \frac{\text{dist}(v_i, v_{r,k})}{\text{vel}(r)}$$

where $\text{vel}(r)$ is the estimation of average velocity of the r-th robot, which can be revised while patrolling, and $\text{dist}(v_i, v_{r,k})$ is the path length between vertex v_i and $v_{r,k}$. The path length $\text{dist}(v_i, v_{r,k})$ can be expressed as the weight $c_{i,r}$ of the edge connecting vertex $v_i, v_{r,k} \in V$. In order to avoid visiting several close vertices repeatedly, a lower bound should be set to $dist(v_i, v_{r,k})$. Then

$$\text{dist}(v_i, v_{r,k}) = \max\{c_{i,r}, c_G^{avg}\} \tag{5}$$

where c_G^{avg} is the average weight of edges in the graph and used as the lower bound.

The expected time of arrival at vertex v_i is estimated according to the current time t and the travel time Δt, as

$$t_{next} = t + \Delta t$$

The expected idleness is used instead of the instantaneous idleness to evaluate the urgency of visiting the vertex. If the vertex v_i belongs to the set S_{ints}, which means the vertex is expected to be visited by another robot at time t_{exp}, the expected idleness is calculated as

$$I_{v_i}^{exp} = t_{next} - t_{exp}$$

Otherwise, the calculation of expected idleness is similar to instantaneous idleness as

$$I_{v_i}^{exp} = t_{next} - t_{last}$$

where t_{last} is the last time that the vertex was visited.

The travel time Δt is used instead of the path length $dist(v_i, v_{r,k})$ to represent the cost function. The utility of visiting vertex v_i is judged as

$$U(v_i) = |I_{v_i}^{exp}| / \Delta t \tag{6}$$

For each of the adjacent vertex $v_i \in N(v_{r,k})$, the utility function $U(v_i)$ is calculated. The vertex with the maximum utility is chosen as the next target vertex $v_{r,k+1}$.

After the robot determines its next target vertex, it publishes a message of intention $(v_{r,k+1}, t_{next})$, which contains the expected time of arrival at the next vertex.

The process of the ER algorithm is reviewed in algorithm 1. This algorithm is executed on each robot, respectively, and the decision process of each robot is asynchronous and independent of each other. The cooperation between robots can be achieved through the messages of arrival and intention, whose structures are tuples (v_*, t_*) consisting of vertex id and time instant.

Algorithm 1 ER Algorithm

```
1:  procedure UPON REACH VERTEX v_{r,k} AT TIME t
2:      Publish message of arrival (v_{r,k}, t);
3:      Update S_ints; // the set which contains the intentions of
            other robots.
4:      // Choose vertex to visit next from neighbours of v_{r,k}.
5:      for all v_i ∈ N(v_{r,k}) do
6:          Δt = dist(v_i, v_{r,k}) / vel(r);
7:          t_next = t + Δt;
8:          if v_i ∈ S_ints then
9:              I^exp_{v_i} = t_next − t_exp;
10:         else
11:             I^exp_{v_i} = t_next − t_last;
12:         U(v_i) = |I^exp_{v_i}| / Δt;
13:     v_{r,k+1} ← arg max{U(v_i)};
14:     Publish message of intention (v_{r,k+1}, t_next);
15:     Move to vertex v_{r,k+1};

16: procedure UPON RECEIVE MESSAGE OF ARRIVAL(v_i, t'_last)
17:     // Update the last visit time of vertex v_i .
18:     t_last ← max{t_last, t'_last};

19: procedure UPON RECEIVE MESSAGE OF INTENTION(v_i, t'_exp)
20:     // Update the expected visit time of vertex v_i .
21:     if v_i ∉ S_ints then
22:         Add v_i to S_ints;
23:         t_exp ← t'_exp;
24:     else
25:         t_exp ← min{t_exp, t'_exp};
```

Discussion

The meaning of the utility function $U(v_i)$ is discussed in this section.

If the vertex $v_i \notin S_{ints}$ which means there is no robot intending to visit this vertex, the utility function can be simplified as

$$\begin{aligned} U(v_i) &= \frac{|t_{next} - t_{last}|}{\Delta t} \quad ; (t_{next} = t + \Delta t > t_{last}) \\ &= \frac{t + \Delta t - t_{last}}{\Delta t} \\ &= 1 + \frac{t - t_{last}}{\Delta t} \\ &= 1 + \frac{I_{v_i}^{ins}}{\Delta t} \end{aligned}$$

which signifies that the utility function is equal to one plus the instantaneous idleness divided by travel time. Maximizing the utility function means to find vertex left unvisited for a long time and also can be reached quickly.

If the vertex $v_i \in S_{ints}$ which means the vertex is expected to be visited by another robot at time t_{exp}, the utility function can be simplified as

$$\begin{aligned} U(v_i) &= \frac{|t_{next} - t_{exp}|}{\Delta t} \quad ; (t_{exp} = t_{last} + \Delta t') \\ &= \frac{|t + \Delta t - t_{last} - \Delta t'|}{\Delta t} \\ &= \left| 1 + \frac{t - t_{last}}{\Delta t} - \frac{\Delta t'}{\Delta t} \right| \\ &= \left| 1 + \frac{I_{v_i}^{ins}}{\Delta t} - \frac{\Delta t'}{\Delta t} \right| \end{aligned}$$

where $\Delta t'$ is the travel time of another robot moving to vertex v_i. The absolute value is taken in the utility function, because t_{exp} may be greater than t_{next}, which means the robot may arrive at vertex v_i earlier than other robots. The intention of other robots visiting vertex v_i introduces a penalty term in the utility function, so as to decrease the utility of competing for the same vertex. In this case, the robot tends to choose other free vertex or vertex with larger time interval between arrivals, so as to decrease the occurrence of conflicts between robots.

Experiments

Both simulation experiments and real robot experiments are used to verify the validity and applicability of the proposed algorithm in this article. Simulation experiments are implemented under the 2-D multi-robot simulator, Stage,[25] together with the Robot Operating System (ROS) environment. Three TurtleBot (http://www.turtlebot.com/) robots driven by ROS[26] are utilized to perform the patrol algorithm in a corridor environment. With the convenience brought by ROS, the code can run both on real and simulated robots by modifying a few configuration parameters (the code is available at https://github.com/Chuanbo/my_patrol_sim).

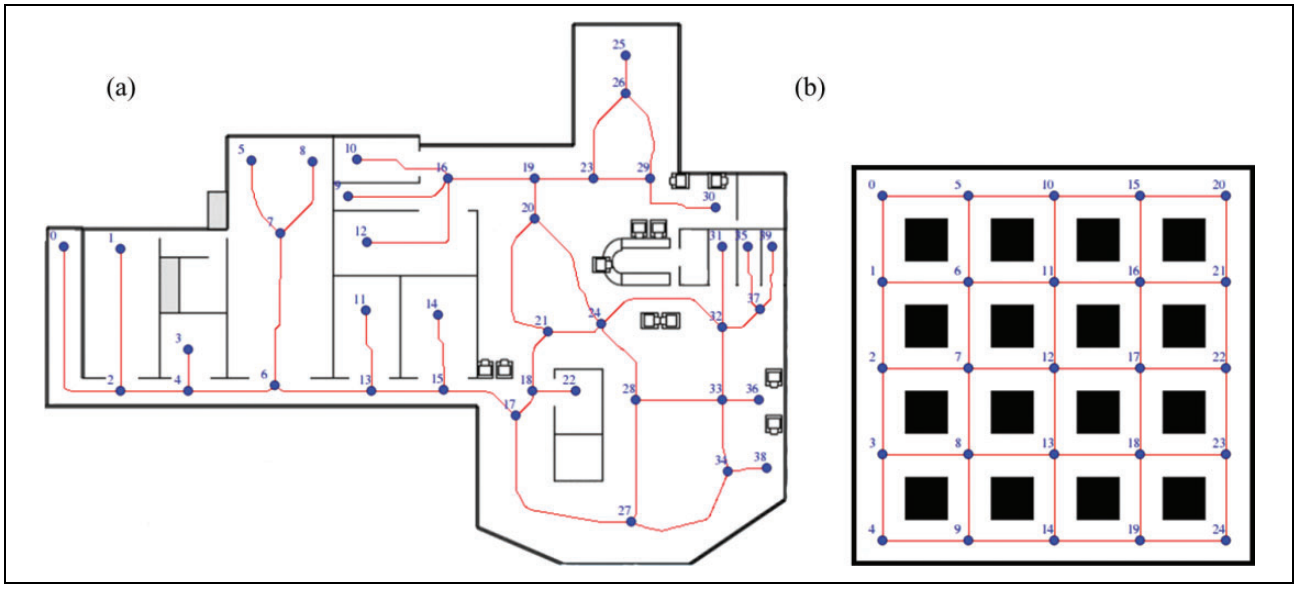

Figure 1. Maps with different graph structures used in simulation experiments. (a) cumberland and (b) grid.

Simulation experiments

Stage simulates mobile robots with different kinematics models and sensors in a 2-D bitmapped environment. The simulated robot is differential drive with a maximum velocity of 0.3 m/s and carries a range sensor with 60° forward-facing horizontal field of view (detection range: 0–10 m)The configuration of the simulated robot is similar to that of the TurtleBot robot.

Two maps (reference from https://github.com/davidbsp/patrolling_sim) with different graph structures are used to evaluate the performance of related patrol algorithms, as shown in Figure 1. The graph information used for patrolling is obtained in advance and known by all the robots. The map *cumberland* is constructed from an indoor environment with 40 vertices and 44 edges. The map *grid* is a virtual environment with 25 vertices and 40 edges. The algebraic connectivity (the algebraic connectivity of a graph is the second smallest eigenvalue of the normalized Laplacian matrix, and it is greater than 0 for a connected graph) of *cumberland* and *grid* is 0.0109 and 0.1313, respectively, which means *grid* is better connected than *cumberland*.

A monitoring program has been implemented with two purposes. First, it visualizes the state of robots and controls the start as well as stop of the simulation (Figure 2). Second, it listens to the publishing messages from patrolling robots to record the idleness information. The idleness of each vertex begins to be recorded since the second visit by a robot, so as to skip the transition period and reach the steady patrol state. The estimation of average velocity $vel(r)$ is chosen to be 0.2 m/s in the simulation.Comparative simulations between the proposed algorithm –ER and several other patrol algorithms were conducted in the two maps with different numbers of robots (1, 4, 8, 12). The duration of the simulations is ranged from 1 h to 8 h according to the number of robots (about 30 visits per vertex), and each simulation was repeated three times. The related algorithms involve two distributed algorithms: SEBS,[10] CR[17] and a centralized partition (CP)-based algorithm. The CP-based algorithm uses program METIS (https://mathema.tician.de/software/pymetis)[13] to partition the graph and computes the local patrolling route with a greedy TSP solver (https://github.com/dmishin/tsp-solver). It should be noted that this centralized algorithm is not fully optimized, as the focus of this article is the distributed algorithm. The average idleness of the graph I_G^{avg}, the maximum average idleness of all vertices $\max\{I_V^{avg}\}$ and the standard deviation σ are calculated (in seconds) as the evaluation criteria of different algorithms. The simulation results are presented in Tables 1 and 2. Since each simulation is repeated three times, there are three results in each case. Among the comparable results in each row, the best three results are indicated in bold font.

In the case of single robot patrolling, the decision rule of ER is similar to that of SEBS (since the distribution function $F(g)$ in SEBS is monotonically increasing,[10] the probabilistic model dose not affect the decision of choosing vertex with the largest utility function), but the constraint condition equation (5) of path length between vertices is different. Compared with SEBS, the proposed algorithm (ER) gives results with higher average idleness I_G^{avg} but lower $\max\{I_V^{avg}\}$ and σ. The performance of other algorithms is similar, because there is no conflict between robots when using only one robot.

The difference in performance of these algorithms appears gradually as the number of robots increases. In the case of multi-robot patrolling, ER and SEBS perform better than other distributed algorithms and can even have a comparable with the CP-based algorithm. In other distributed algorithms, conflicts between robots increase with the

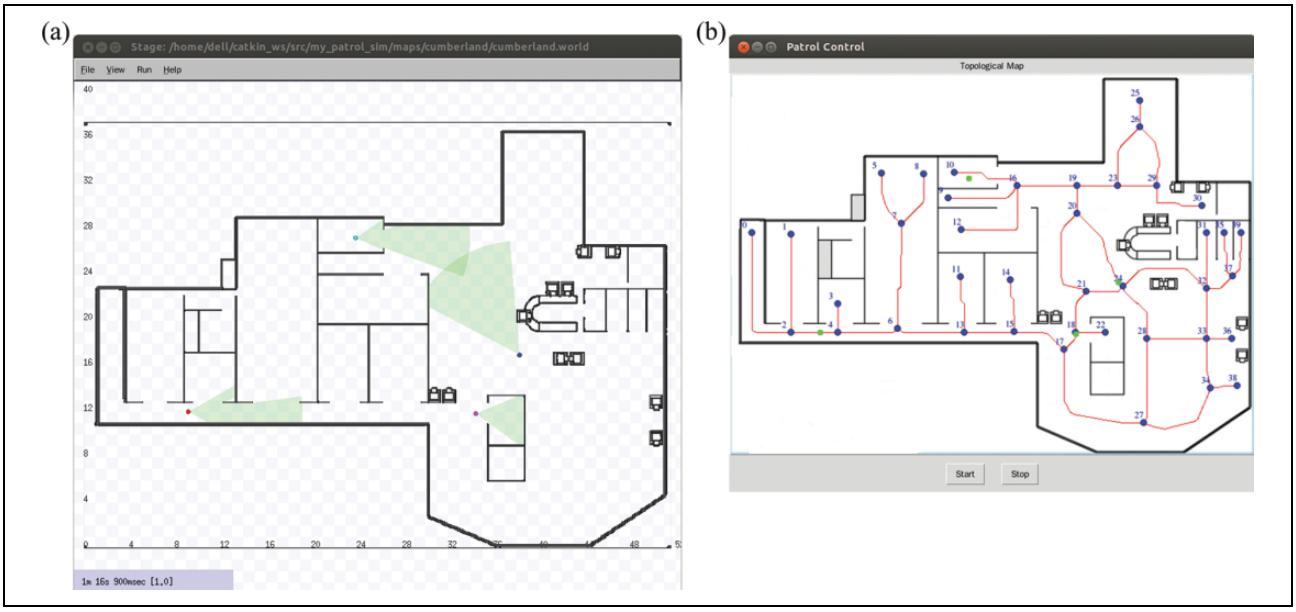

Figure 2. The Stage simulator and the monitor with map cumberland. (a) The Stage simulator and (b) the monitor.

Table 1. Simulation results in map cumberland.

Number	Criterion	ER			SEBS			CR			CP		
1	l_G^{avg}	1029.1	1034.8	1030.0	995.9	1000.8	995.8	1033.4	1040.8	1022.1	**980.2**	**982.7**	**980.0**
	$\max\{l_V^{avg}\}$	1648.8	1638.7	1641.1	2939.3	2938.2	2938.5	1672.2	1660.0	1628.9	**1403.2**	**1403.1**	**1402.9**
	σ	512.6	512.4	515.4	543.0	542.8	542.6	514.6	521.3	502.1	**437.4**	**434.6**	**437.3**
4	l_G^{avg}	242.8	244.6	**239.1**	**230.5**	**230.1**	243.9	373.3	258.7	282.6	243.2	243.1	243.4
	$\max\{l_V^{avg}\}$	**398.4**	432.0	**385.7**	528.4	480.7	602.6	773.3	456.9	524.3	401.6	**401.1**	401.3
	σ	**103.8**	**104.6**	**99.8**	112.6	108.9	141.2	195.1	109.8	131.2	116.1	115.8	116.1
8	l_G^{avg}	**111.7**	113.8	115.5	**110.5**	**111.0**	111.8	147.1	150.4	148.0	118.7	118.8	118.7
	$\max\{l_V^{avg}\}$	**199.2**	**178.8**	**174.9**	233.2	246.4	259.4	282.6	323.9	329.6	206.8	206.4	206.6
	σ	**41.7**	**40.7**	**40.9**	49.8	53.8	57.9	67.6	72.0	73.4	51.0	51.0	51.1
12	l_G^{avg}	**72.1**	**72.6**	73.7	77.5	73.4	**72.9**	115.9	109.6	109.9	74.6	74.6	74.7
	$\max\{l_V^{avg}\}$	**141.4**	**140.8**	**144.8**	179.3	175.2	164.6	208.6	203.1	228.4	152.4	152.6	152.3
	σ	**27.2**	**30.3**	**30.8**	35.9	33.7	34.7	46.5	47.9	49.1	36.4	36.5	36.4

ER: expected reactive; SEBS: state exchange Bayesian strategy; CR: conscientious reactive; CP: centralized partition.

Table 2. Simulation results in map grid.

Number	Criterion	ER			SEBS			CR			CP		
1	l_G^{avg}	**516.7**	**516.6**	520.7	520.5	519.0	518.2	**516.8**	523.3	519.8	538.4	538.6	538.8
	$\max\{l_V^{avg}\}$	**540.7**	**533.8**	590.1	579.7	572.0	**538.5**	545.4	587.0	564.4	572.8	572.9	573.2
	σ	**47.7**	**47.9**	78.7	74.4	67.9	52.6	**52.0**	81.4	70.8	95.0	94.6	95.0
4	l_G^{avg}	**127.6**	127.8	128.4	**127.7**	127.8	**127.7**	159.7	157.5	183.8	130.2	130.2	130.1
	$\max\{l_V^{avg}\}$	**157.7**	**157.8**	159.8	160.6	**153.0**	161.2	206.8	198.1	221.7	163.5	163.3	163.1
	σ	12.4	**11.8**	12.8	**11.8**	**12.2**	14.0	21.2	17.8	24.3	20.7	20.7	20.6
8	l_G^{avg}	**63.8**	**64.0**	65.1	64.7	**63.8**	64.2	78.1	78.5	79.4	69.9	69.8	69.8
	$\max\{l_V^{avg}\}$	**72.0**	76.7	84.7	78.6	**72.0**	**73.6**	102.8	117.7	101.9	82.2	82.5	82.9
	σ	4.8	**4.6**	6.2	5.0	**3.9**	**4.5**	8.8	12.3	10.2	18.5	18.5	18.5
12	l_G^{avg}	**43.2**	43.4	**43.3**	43.4	**43.3**	43.4	64.0	62.8	60.1	45.1	45.0	44.9
	$\max\{l_V^{avg}\}$	**46.7**	**46.5**	**46.7**	47.1	47.1	46.8	78.1	79.7	74.9	82.5	82.7	82.6
	σ	**2.0**	**2.0**	2.0	2.1	**2.0**	2.0	5.7	8.0	7.7	11.2	11.4	11.3

ER: expected reactive; SEBS: state exchange Bayesian strategy; CR: conscientious reactive; CP: centralized partition.

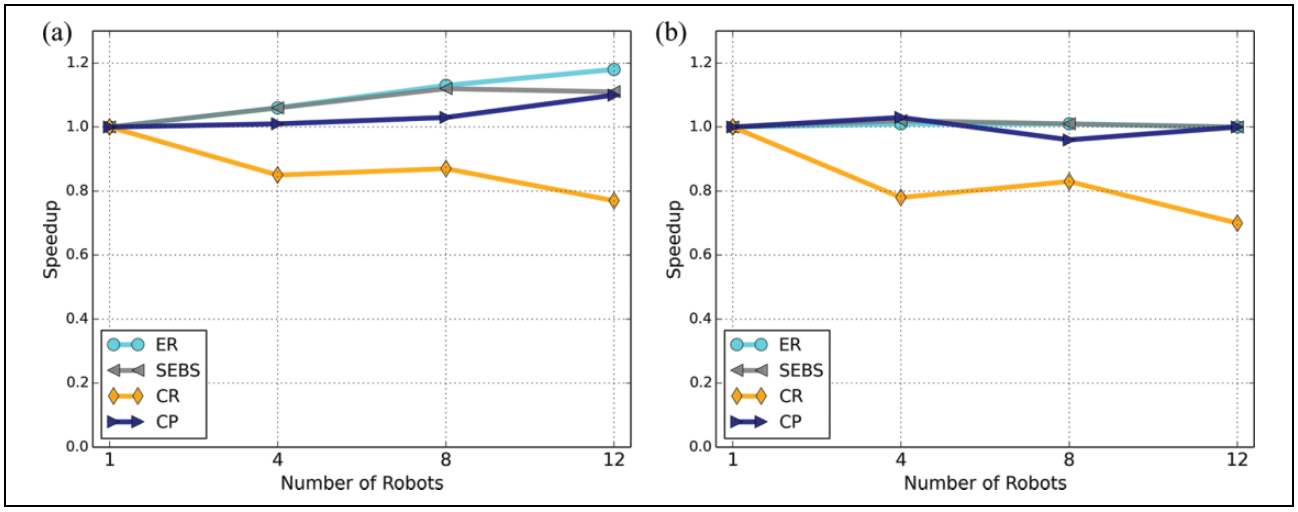

Figure 3. The speed-up of different algorithms (a) in map *cumberland* and (b) in map *grid*.

number of robots. The conflict, which means several robots approach the same region within a short period of time in patrolling, would decrease the efficiency of multi-robot cooperation, as the robot has to avoid collision from each other. In ER and SEBS, conflicts between robots could be reduced by considering the intentions of robots. As a result of graph partition, conflicts can also be avoided in CP.

The speed-up measurement[10,27] is introduced to analyse the scalability of different algorithms and can be calculated as

$$s(r) = \frac{\frac{p(1)}{r}}{p(r)} \tag{7}$$

where $p(r)$ is the performance for r robots measured by average idleness I_G^{avg}. The *speed-up* of each algorithm is compared in Figure 3. In map *cumberland*, the *speed-up* of ER, SEBS and CP is slightly greater than 1, because the task assignment of patrol vertices may prevent robots from moving across long edges which consume more time. Furthermore, the *speed-up* of these three algorithms in map *grid* approximately equals to 1, because all edges have the same length. The *speed-up* of other algorithms in both maps decreases as the number of robots increases.

The results show that ER and SEBS scale well for the number of robots. The distributed characteristic of both algorithms offers the potential of fault tolerance. Compared with the SEBS, the ER reduces the dependence on specific parameters (e.g. parameter L, M and the number of robots R, in SEBS algorithm[10]) and thus has better scalability for dynamically formed robot teams.

Robot experiments

The proposed algorithm in this article has been tested in a *corridor* environment with three TurtleBot robots (Figure 4). A graph with 13 vertices and 14 edges is extracted from the $29.50 \times 11.35\,m$ environment, as shown in Figure 5. Each TurtleBot runs its local ROS master for

Figure 4. TurtleBot robots used in the experiments.

data collection and control of the robot. The lightweight communications and marshalling,[28] a library independent of ROS environment, is used to exchange relevant information between robots. The proposed algorithm (ER) is used in the software layer of each robot to determine the patrol route independently and sequentially.The monitoring program used in simulation is also used in real robot experiments. The average idleness of the graph I_G^{avg}, the maximum average idleness $\max\{I_V^{avg}\}$ and the standard deviation σ of all vertices are calculated (in seconds). The idleness of each vertex began to be recorded since the second visit by a robot, so as to skip the transition period and reach the steady patrol state. The experiment stopped after each vertex had been visited at least four times and was repeated three times. The recorded data is adequate, as the idleness converges in the steady patrol state. The experiment results with one and three robots are presented in Table 3. The values of I_G^{avg}, $\max\{I_V^{avg}\}$ and σ decrease as the number of robots increases. The *speed-up* of I_G^{avg} with three robots $s(3) = 0.96$ is slightly less than 1.

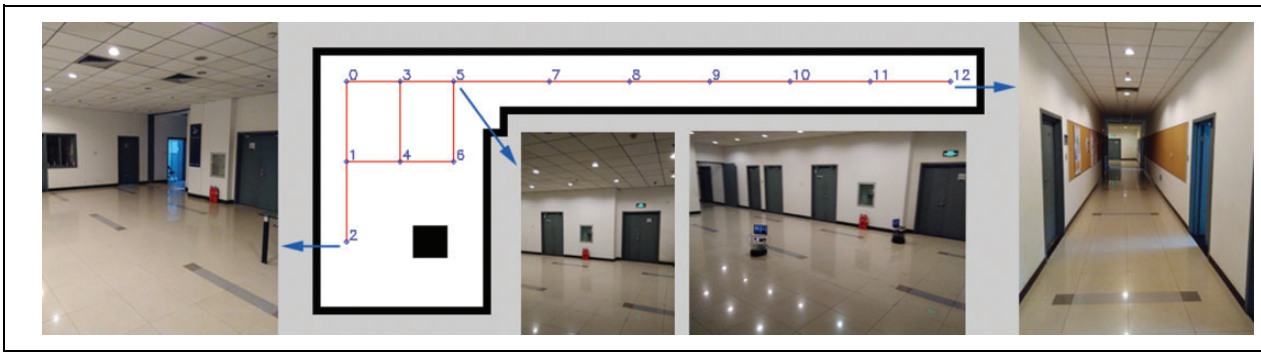

Figure 5. The corridor environment for robot experiments.

Table 3. Experiment results in map corridor.

Number	I_G^{avg}	$\max\{I_V^{avg}\}$	σ
	202.3	278.9	70.9
1	207.7	285.8	72.1
	204.2	281.9	71.3
	72.1	93.8	16.4
3	71.7	99.1	16.1
	69.8	89.9	15.6

In order to verify the convergence of idleness during patrolling, the average value and the maximum value of the instantaneous idleness (defined in equation (4)) are recorded every second and plotted as Figure 6. As the initial idleness is set to 0 for all vertices, there exists a transition period before the average instantaneous idleness $avg\{I_{v_i}^{ins}\}$ reaches stability within a certain scope. It can be seen that the average instantaneous idleness converges quickly after the experiment started. Therefore, the statistic results are adequate to describe the steady state of patrolling. On the other hand, the maximum instantaneous idleness $\max\{I_{v_i}^{ins}\}$ increases linearly with time, until the vertex with the maximum value is visited by a robot; then the curve drops down to the submaximum value and increases linearly again. After the steady state is reached, the maximum idleness is also kept within a certain scope.

Simulation tests about scalability and fault tolerance

In order to further analyse the scalability and fault tolerance of distributed patrol algorithms, more simulation experiments with the Stage simulator were conducted due to the convenience in configuring different test conditions.

The impact of robot failures

The fault tolerance of distributed algorithms is brought by the autonomy of every robot and the redundancy of the robot team. The robot team can be dynamically formed which means robots may join or withdraw during patrolling. This feature is tested through simulations for algorithm ER and SEBS.

Four robots were placed in map *grid* and began patrolling at time 50 s. A robot withdrew from patrolling at time 350 s and rejoined at time 650 s. In order to describe the evolution of the system status, the average value and the maximum value of the instantaneous idleness (defined in equation (4)) are recorded every second and plotted as Figure 7. As the initial idleness is set to 0 for all vertices, there exists a transition period before the idleness reaches stability within a certain scope. After a robot quit, the idleness rose to a higher level, but the patrolling team remained functional with the robots left (the idleness of every vertex remained within a limited scope). Furthermore, the idleness fell back to normal soon after the stopped robot rejoined the team. So it can be seen that both ER and SEBS are robust to robot failures and tolerate the robot to join or withdraw during patrolling.

The impact of unreliable communications

In the previous experiments, communication between robots is assumed to be reliable, which means the robot can receive all messages sent between each other. But in practice, this is not always the case. Messages between robots may be lost or delayed because of unreliable communications and may even be confined to local interactions because of restrictions on communication range.

Comparative simulations on different error rates of communication were conducted in map *grid* with eight robots for algorithm ER and SEBS. In the simulations, the robot would ignore the received message with a certain probability: 0, 25, 50 and 75%, so as to simulate different error rates. Each simulation lasted 1 h (about 30 visits per vertex) and was repeated three times. The average idleness of the graph I_G^{avg}, the maximum average idleness of all vertices $\max\{I_V^{avg}\}$ and the standard deviation σ are calculated (in seconds) to evaluate the results, as shown in Table 4. The performance decreases as the error rate of communication increases. The failures in message delivery would lead to incomplete representation of idleness and intention information in all the robots. The robots might choose

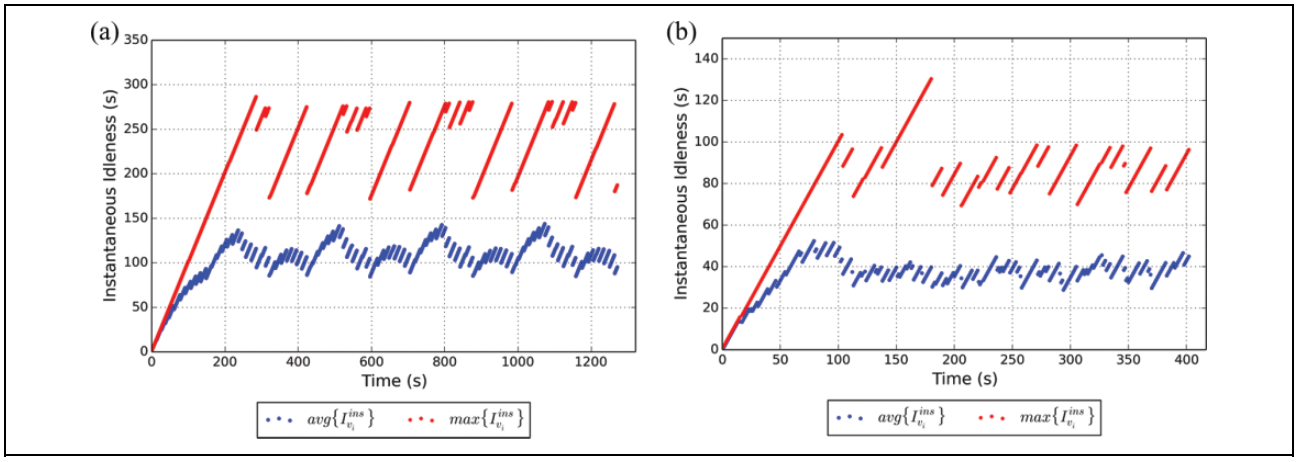

Figure 6. The instantaneous idleness along time. (a) one robot and (b) three robots.

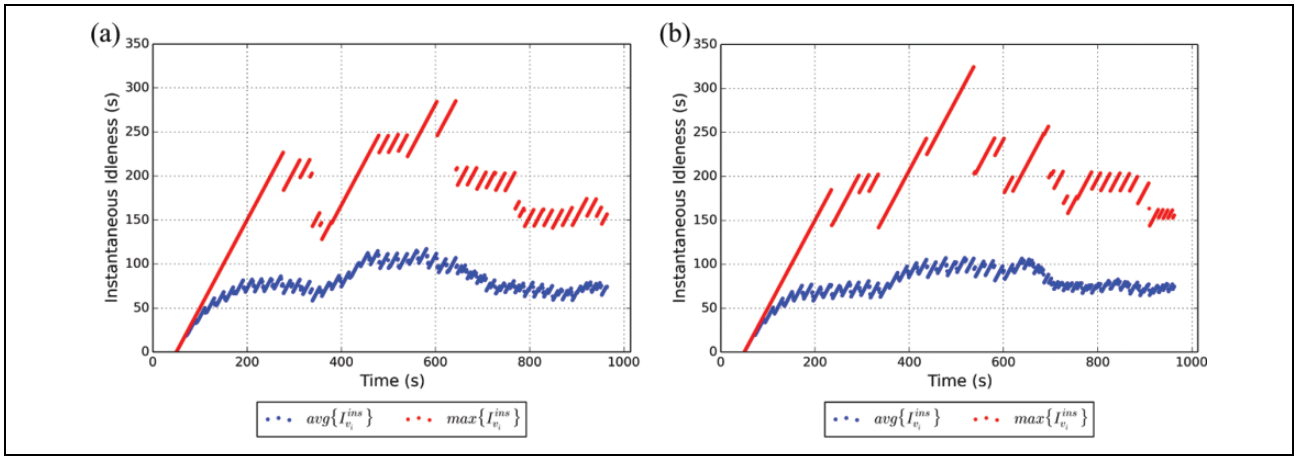

Figure 7. The instantaneous idleness along time with robot failures. (a) ER and (b) SEBS. ER: expected reactive; SEBS: state exchange Bayesian strategy.

suboptimal plans or come into conflicts, as the result of their incomplete information. But it is worth noting that the performance degradation is only 11% when the error rate is 25%, which means these distributed algorithms could still perform well under communication errors within a certain scope.

Similar to the preceding configuration, comparative simulations on the time delay of communication were conducted as well. The difference is that the robot would wait for a certain time: (0 s, 5 s, 10 s and 15 s) before processing the received message in order to simulate the time delay. Note that the time required by a robot to move between two adjacent vertices in map *grid* is about 20 s. The simulation results are shown in Table 5. The delay in message delivery reduces the performance of the distributed algorithms. But the performance degradation is only 9% when the delay is 5, which means these algorithms could tolerate moderate communication delay and still perform well.

Simulations about the restriction on communication range were conducted in map *grid* with eight robots for algorithm ER and SEBS. In the simulations, the robot would only receive messages from robots whose distance to the robot is less than 12 m, so as to permit only local communication between robots within two edges in the

Table 4. Performance with different error rates of communication.

Rate	Criterion	ER			SEBS		
0%	I_G^{avg}	63.8	64.0	65.1	64.7	63.8	64.2
	$\max\{I_V^{avg}\}$	72.0	76.7	84.7	78.6	72.0	73.6
	σ	4.8	4.6	6.2	5.0	3.9	4.5
25%	I_G^{avg}	77.9	72.2	68.8	71.5	70.7	68.3
	$\max\{I_V^{avg}\}$	107.9	97.8	81.7	103.2	93.6	82.6
	σ	12.0	9.2	6.1	9.4	9.6	6.8
50%	I_G^{avg}	76.3	75.1	76.7	73.9	91.8	77.3
	$\max\{I_V^{avg}\}$	116.7	107.8	104.9	106.6	126.3	101.6
	σ	11.7	11.1	10.3	12.1	17.1	10.6
75%	I_G^{avg}	80.5	82.4	80.4	79.9	76.7	89.0
	$\max\{I_V^{avg}\}$	104.3	117.4	105.9	115.5	98.8	117.3
	σ	8.6	14.5	11.0	10.8	9.3	13.7

ER: expected reactive; SEBS: state exchange Bayesian strategy.

Table 5. Performance with time delay of communication.

Delay	Criterion	ER			SEBS		
0 s	I_G^{avg}	63.8	64.0	65.1	64.7	63.8	64.2
	$\max\{I_V^{avg}\}$	72.0	76.7	84.7	78.6	72.0	73.6
	σ	4.8	4.6	6.2	5.0	3.9	4.5
5 s	I_G^{avg}	74.2	69.6	68.4	70.2	69.6	67.0
	$\max\{I_V^{avg}\}$	108.0	86.3	90.0	88.0	92.4	77.9
	σ	10.9	7.3	6.9	7.3	7.2	5.3
10 s	I_G^{avg}	77.8	75.6	73.6	95.7	80.7	73.8
	$\max\{I_V^{avg}\}$	110.1	97.9	90.8	141.4	104.9	89.8
	σ	10.4	10.3	8.7	17.1	8.0	10.0
15 s	I_G^{avg}	80.8	78.1	80.2	106.6	80.0	86.5
	$\max\{I_V^{avg}\}$	110.6	94.9	116.1	146.5	98.3	134.4
	σ	11.1	8.3	11.6	14.4	8.1	16.3

ER: expected reactive; SEBS: state exchange Bayesian strategy.

Table 6. Performance with restriction on communication range.

Range	Criterion	ER			SEBS		
Global	I_G^{avg}	63.8	64.0	65.1	64.7	63.8	64.2
	$\max\{I_V^{avg}\}$	72.0	76.7	84.7	78.6	72.0	73.6
	σ	4.8	4.6	6.2	5.0	3.9	4.5
12 m (two edges)	I_G^{avg}	65.0	66.8	68.4	70.7	72.1	68.8
	$\max\{I_V^{avg}\}$	83.8	92.6	81.7	80.5	90.0	88.8
	σ	6.6	8.5	7.3	5.3	8.9	7.1
6 m (one edge)	I_G^{avg}	89.4	85.3	87.5	91.1	85.1	94.6
	$\max\{I_V^{avg}\}$	137.7	113.5	120.5	131.5	99.2	115.5
	σ	14.1	11.0	11.9	17.0	9.6	12.0

ER: expected reactive; SEBS: state exchange Bayesian strategy.

graph. Since the robot makes decisions and publishes messages at the location of vertex in the graph, it is difficult to avoid conflicts when the communication range is less than two edges. But for purposes of comparison, the simulations with a smaller communication range – 6 m (one edge in the graph) – were also conducted. Each simulation lasted 1 h (about 30 visits per vertex) and was repeated three times. The simulation results (I_G^{avg}, $\max\{I_V^{avg}\}$ and σ) compared with the case of global communication between robots are shown in Table 6. In the case of range 6 m (one edge in the graph), the smaller communication range significantly degrades the performance of both ER and SEBS due to the aforementioned reason. The performance is even worse than in the case of 75% error rate of communication (Table 4). In contrast, range 12 m (two edges in the graph) ensures the local communication between adjacent robots, and only causes performance degradation 4% for ER and 10% for SEBS. As expected, these two distributed algorithms are applicable to robot team with only local communication. Since both algorithms are reactive rather than cognitive (the robot only determines the next vertex to visit from neighbors of the current vertex), the impact of incomplete information about distant vertices is not significant. But the restriction on communication range indirectly affects the validity of the parameters based on the number of robots for the SEBS. In contrast, the ER doesn't require such parameters and thus performs better.

Conclusions

In this article, focusing on the frequency-based patrol with multi-robot, a distributed algorithm based on expected idleness is proposed. Using the information shared between robots, the expected idleness is estimated by each robot, which indicates the impact of other robots. In the decision process, the expectation about future status is considered to avoid potential conflicts between robots. The proposed ER algorithm is adaptive and distributed without the need for off-line planning or centralized control.

Comparisons with the state-of-the-art algorithms have been conducted in a realistic simulator, Stage, and shown the ER algorithm is efficient in patrolling which could even be comparable with centralized algorithms. More simulations have also been conducted to verify the scalability and fault tolerance. Experiments on real robots have further verified the validity and applicability of the ER algorithm. Further research to improve the algorithm for weighted idleness, which represents different priority of each vertex will be future works.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

References

1. Guo Y, Parker LE and Madhavan R. Collaborative robots for infrastructure security applications. In: Nedjah N, dos Santos Coelho L and de Macedo Mourelle L (eds) *Mobile robots: the evolutionary approach*. Berlin/Heidelberg: Springer, 2007, pp. 185–200.
2. Cabrita G, Sousa P, Marques L, et al. Infrastructure monitoring with multi-robot teams. In: *2010IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 18–22. IEEE.
3. Agmon N, Kaminka GA and Kraus S. Multi-robot adversarial patrolling: facing a full-knowledge opponent. *J Artif Intell Res* 2011; 42: 887–916.
4. Elmaliach Y, Agmon N and Kaminka GA. Multi-robot area patrol under frequency constraints. *Ann Math Artif Intell* 2009; 57(3–4): 293–320.
5. Arkin EM, Hassin R and Levin A. Approximations for minimum and min-max vehicle routing problems. *J Algorithms* 2006; 59(1): 1–18.

6. Bektas T. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega* 2006; 34(3): 209–219.
7. Pasqualetti F, Franchi A and Bullo F. On cooperative patrolling: optimal trajectories, complexity analysis, and approximation algorithms. *IEEE Trans Robot* 2012; 28(3): 592–606.
8. Chevaleyre Y. Theoretical analysis of the multi-agent patrolling problem. In: *2004 IEEE/WIC/ACM international conference on intelligent agent technology (IAT)*, 2004. pp 302–308. IEEE.
9. Portugal D, Pippin C, Rocha RP, et al. Finding optimal routes for multi-robot patrolling in generic graphs. In: *2014 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2014, pp. 363–369. IEEE.
10. Portugal D and Rocha RP. Distributed multi-robot patrol: a scalable and fault-tolerant framework. *Robot Auton Syst* 2013; 61(12): 1572–1587.
11. Agmon N, Fok CL, Emaliah Y, et al. On coordination in practical multi-robot patrol. In: *2012 IEEE international conference on robotics and automation (ICRA)*, 2012, pp. 650–656. IEEE.
12. Portugal D and Rocha R. MSP algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning. In: *2010 ACM symposium on applied computing (SAC)*, 2010, pp. 1271–1276. ACM.
13. Karypis G and Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J Sci Comput* 1998; 20(1): 359–392.
14. Portugal D and Rocha RP. Multi-robot patrolling algorithms: examining performance and scalability. *Adv Robot* 2013; 27(5): 325–336.
15. Pippin C, Christensen H and Weiss L. Performance-based task assignment in multi-robot patrolling. In: *28th Annual ACM symposium on applied computing*, ACM, 2013, pp. 70–76.
16. Pasqualetti F, Durham JW and Bullo F. Cooperative patrolling via weighted tours: performance analysis and distributed algorithms. *IEEE Trans Robot* 2012; 28(5): 1181–1188.
17. Machado A, Ramalho G, Zucker JD, et al. Multi-agent patrolling: an empirical analysis of alternative architectures. In: *3rd international workshop on multi-agent-based simulation*, 2002, pp. 155–170. Berlin/Heidelberg: Springer.
18. Almeida A, Ramalho G, Santana H, et al. Recent advances on multi-agent patrolling. In: *Advances in artificial intelligence–SBIA 2004, 17th Brazilian symposium on artificial intelligence SBIA*, 2004, pp. 474–483. Berlin/Heidelberg: Springer.
19. Menezes T, Tedesco P and Ramalho G. Negotiator agents for the patrolling task. In: *Advances in artificial intelligence – IBERAMIA-SBIA*, 2006, pp. 48–57. Berlin/Heidelberg: Springer.
20. Poulet C, Corruble V and Seghrouchni AEF. Auction-based strategies for the open-system patrolling task. In: *15th international conference on principles and practice of multi-agent systems (PRIMA)*, 2012, pp. 92–106. Berlin/Heidelberg: Springer.
21. Hernandez ABE and DelCerro J. Game theory models for multi-robot patrolling of infrastructures. *Int J Adv Robot Syst* 2013; 10(2): 463–474.
22. Iocchi L, Marchetti L and Nardi D. Multi-robot patrolling with coordinated behaviours in realistic environments. In: *2011 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2011, pp. 2796–2801. IEEE.
23. Beeson P, Jong NK and Kuipers B. Towards autonomous topological place detection using the extended Voronoi graph. In: *2005 IEEE international conference on robotics and automation (ICRA)*, 2005, pp. 4373–4379. IEEE.
24. Portugal D and Rocha RP. Extracting topological information from grid maps for robot navigation. In: *4th international conference on agents and artificial intelligence (ICAART)*, 2012, pp. 137–143. SciTePress.
25. Gerkey B, Vaughan RT and Howard A. The player/stage project: tools for multi-robot and distributed sensor systems. In: *11th international conference on advanced robotics*, 2003, pp. 317–323.
26. Quigley M, Conley K, Gerkey B, et al. ROS: an open-source robot operating system. In: *ICRA workshop on open source software*, 2009, p. 5. IEEE.
27. Balch T and Arkin RC. Communication in reactive multiagent robotic systems. *Auton Robot* 1994; 1(1): 27–52.
28. Huang AS, Olson E and Moore DC. LCM: lightweight communications and marshalling. In: *2010 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2010, pp. 4057–4062. IEEE.