Deployment of Mobile Robots With Energy and Timing Constraints

Yongguo Mei, *Student Member, IEEE*, Yung-Hsiang Lu, *Member, IEEE*, Y. Charlie Hu, *Member, IEEE*, and C. S. George Lee, *Fellow, IEEE*

***Abstract*—Mobile robots can be used in many applications, such as carpet cleaning, search and rescue, and exploration. Many studies have been devoted to the control, sensing, and communication of robots. However, the deployment of robots has not been fully addressed. The deployment problem is to determine the number of groups unloaded by a carrier, the number of robots in each group, and the initial locations of those robots. This paper investigates robot deployment for coverage tasks. Both timing and energy constraints are considered; the robots carry limited energy and need to finish the tasks before deadlines. We build power models for mobile robots and calculate the robots' power consumption at different speeds. A speed-management method is proposed to decide the traveling speeds to maximize the traveling distance under both energy and timing constraints. Our method uses rectangle scanlines as the coverage routes, and solves the deployment problem using fewer robots. Finally, we provide an approach to consider areas with random obstacles. Compared with two simple heuristics, our solution uses 36% fewer robots for open areas and 32% fewer robots for areas with obstacles.**

***Index Terms*—Coverage, deployment, energy constraints, mobile robots, timing constraints.**

I. Introduction

MOBILE robots can be used in many applications, such as carpet cleaning, lawn mowing, hazard detection, and exploration of unknown areas. Multiple robots may work collectively to accomplish a common task, for example, searching and rescuing survivors in an urban area after an earthquake. Existing studies about mobile robots focus mostly on enhancing individual robots' capability, such as sensing, obstacle detection and avoidance, localization, motion planning, or interactions with human controllers. Few studies have been conducted for the initial deployment of mobile robots. In this paper, robots are transported by large carriers, such as trucks, from where the robots are stored into working fields. The deployment problem is to decide: 1) the number of groups unloaded by the carriers; 2) the number of robots in each group; and 3) the initial location of each group. After deployment, the robots execute coverage tasks. Coverage is a common problem for search, exploration, and cleaning. In the deployment problem, two important issues need to be considered: energy constraints and timing constraints. Energy constraints are critical to mobile robots because they usually carry batteries with limited capacity. For instance, a Honda humanoid robot can walk for only 30 min with a battery pack [2]. Makimoto *et al.* [28] predict that "the robot will provide the biggest challenges for the low power electronics in the future." Meanwhile, many tasks have timing constraints. For example, after a disaster, survivors usually need to be rescued within 24 hrs; otherwise, the chance of survival diminishes rapidly. Another example is to detect and destroy landmines before troops arrive. Energy constraints and timing constraints can be conflicting optimization goals. It is well known that vehicle fuel efficiency (km per liter) drops dramatically at very high speeds. In other words, meeting the timing constraints by moving at a high speed may reduce energy efficiency. Consequently, it is crucial to consider both energy and timing constraints together.

We can use survivor detection after an earthquake as an example to explain the deployment problem. Mobile robots can move under rubble and find survivors. Each robot is equipped with sensors to detect survivors. When a survivor is found, the robot sends wireless signals to inform rescuers. Before an earthquake, these robots are stored in an emergency response center. After an earthquake, the robots are transported by a carrier to the earthquake site and help rescuers find survivors. Deploying robots is the process of transporting the robots into the field, and robots moving (scattering) from the unloading location to their individual starting locations for coverage. The unloading time and robot scattering time are the overheads of deployment. The deployment problem is to decide the groups of robots unloaded by the carrier and their initial locations. The answers to these questions are affected by each robot's energy capacity, the deadline, the moving speed, and the obstacles. A desirable deployment strategy should meet the following goals: 1) it uses the minimum number of robots to cover a given area; namely, it can cover the maximum area with the same number of robots; 2) it can cover the area within the energy and the timing constraints.

Robot deployment is a complex problem. The traveling speeds of robots are an important factor in deployment. If the robots travel faster, they consume more power but can finish the task sooner. We build a power model for mobile robots, including the power of motors, sensors, and microcontrollers.

Manuscript received April 1, 2005; revised October 22, 2005. This paper was recommended for publication by Associate Editor L. Parker and Editor H. Arai upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant IIS-0329061. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation. This paper was presented in part at the IEEE International Conference on Robotics and Automation, New Orleans, LA, 2004; in part at the IEEE International Conference on Robotics and Automation, Barcelona, Spain, 2005; in part at the International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2005; and in part at the International Conference on Advanced Robotics, Seattle, WA, 2005.

The authors are with School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: ymei@purdue.edu; yunglu@purdue.edu; ychu@purdue.edu; csglee@purdue.edu).

Digital Object Identifier 10.1109/TRO.2006.875494

With the power model, we propose a speed-management method that can maximize the traveling distance under energy and timing constraints. We consider three kinds of deployment overhead: unloading time; dispersing time; and overlapped area. An efficient deployment algorithm can reduce the overhead and fleet size (number of robots used) to cover the same area. Three kinds of overhead are balanced to find an efficient solution to this problem. For example, when we increase the group size (number of robots in each group) and reduce the number of groups, unloading time can be saved, but the dispersing time increases to cover a larger area. We develop an algorithm named the Space Partition Area Coverage Algorithm (SPACA); it determines the groups of robots and their initial locations for minimizing total deployment overhead. Two probability models are developed to describe obstacles in the environment. The first models scattered obstacles, and the second models clustered obstacles. We derive an empirical rule to calculate the additional distance each robot needs to travel for detouring around obstacles. The rule is validated through simulations. Our approach is compared with two other deployment strategies. One strategy unloads all robots at the same location. The second strategy unloads the same number of robots in multiple locations. Our method uses 36% fewer robots than these two strategies in open areas, and 32% fewer in areas with obstacles.

This paper has three major contributions: 1) it develops energy models of mobile robots considering multiple components and provides a speed-management method; 2) it presents an efficient deployment algorithm to reduce the overhead and the fleet size; and 3) it demonstrates an approach to adjust the deployment strategy in environments with obstacles. The rest of this paper is arranged as follows. We first introduce some related work in Section II, and then formally define the robot deployment problem in Section III. Power models for two different robots and a speed-management method are developed in Section IV. Section V presents our deployment strategies. In Section VI, we provide an approach to handle environments with obstacles. After presenting simulation results in Section VII, the paper is concluded in Section VIII.

II. Related Work

Mobile robots can be used in many applications. Carpet cleaning [24] and entertainment [21] are two kinds of applications that we may encounter at our homes. Mobile robots are also used for pickup and delivery tasks [17]. Multiple robots can work together to accomplish a task, and communication among them is important for cooperation purposes. Das *et al.* [11] evaluate three communication protocols in supporting many-to-one communication for mobile robots. Rybski *et al.* use small robots for reconnaissance and surveillance [37]. Exploration using mobile robots has been studied by many researchers. Zelinsky [48] uses a quad-tree data structure to model the environment and presents an adaptive path-planning algorithm for a robot exploring an unknown environment. Batalin *et al.* [5] present an algorithm to cover an area using markers. Their algorithm does not require localization. Gonzalez *et al.* [19] design a coverage algorithm combining spiral paths and backtracking to completely cover an area. Taylor *et al.* [43] model the environment by boundary graphs and present their vision-based path-planning algorithms. Mobile robots can also be used in extraterrestrial explorations. Matthies *et al.* [29] build a small rover as a testbed for Mars exploration. They discuss localization, obstacle detection, and path planning; they also evaluate the performance of the robot executing these operations. In recent years, many researchers investigate simultaneous localization and mapping (SLAM) using mobile robots. SLAM is a problem closely related to exploration. Dellaert *et al.* [13] present a linear algorithm to detect 2-D structures and motions; this algorithm provides initial estimates for multirobot SLAM. Chang *et al.* [7] use a logarithmic map-partition method to reduce the computational complexity for SLAM. Mobile robots are also used for search and rescue applications. Davids *et al.* [12] introduce the application of mobile robots in search and rescue in urban areas. Schreiner *et al.* [38] discuss the research issues in landmine detection using robots. Tadokoro *et al.* [42] investigate the requirements for rescue robots based upon an analysis of an earthquake which happened in Kobe, Japan. Baltes *et al.* [3] demonstrate a binary space-partition method for robot rescue; this method is useful for path planning in a dynamic environment. Zhang *et al.* [49] use a probabilistic method in searching landmines. Their algorithm computes probabilistic distributions of the landmine locations, and uses the distributions to direct the robot's search.

Mobile robots usually carry rechargeable batteries; therefore, energy conservation is important. A mobile robot has several major components: sensors; motors; microcontrollers; and computers. Some studies concentrate on motion planning to reduce the motor's power consumption. Sun *et al.* [41] find energy-efficient paths using topography information of the ground. We present an energy model for mobile robots and discuss the energy properties of three different coverage methods: scanline; spiral; and square spiral [32]. Barili *et al.* [4] describe the concept of controlling the velocities to save energy for a mobile robot. Their work does not consider the relationship between path planning and velocity control. Katoh *et al.* [22] demonstrate an approach for energy conservation by creating elliptic paths and focus on space manipulators for flying robots. Duleba *et al.* [15] discuss nonholonomic energy-efficient motion planning based on the Newton algorithm. Yamasaki *et al.* [47] develop control algorithms to reduce the motion power of humanoid robots. Some studies analyze energy breakdowns for mobile robots [34], [35]; they investigate the power of sensors, controllers, and communication. However, little research focuses on energy-conservation techniques for these components. One of them is the paper by Liu *et al.* [25] on power-aware scheduling algorithms for a Mars rover. Their method considers free solar energy and schedules the rover's operations to use the solar energy and battery efficiently.

The deployment problem is important to multirobot applications. However, only a few existing studies discuss the deployment problem. Simmons *et al.* [39] study coordination techniques among a group of robots. They demonstrate the effectiveness of their method by deploying robots from the same location to their individual destinations. Their paper focuses on control and coordination. Rybski *et al.* [37] use large "ranger" robots to transport and deploy small scout robots. The rangers can travel up to 20 km, greatly extending the search

range of scouts. Chang *et al.* [8] study the energy and time properties of different dispatching algorithms for ant-like robot systems. Yamaguchi [46] discusses adaptive formation control for mobile robots to keep their relative positions. A related problem is the fleet-size problem, which determines the number of robots needed for a task. An efficient deployment strategy can reduce the overhead and use fewer robots to accomplish the same task. Our earlier work presents a probabilistic model to determine the fleet size to satisfactorily serve requests with timing and energy constraints [31]. The requests are modeled by random processes. However, the paper does not consider the initial deployment. One of our previous works [33] presents an efficient deployment strategy to reduce the overhead and the fleet size for areas without obstacles.

In recent years, several studies have been conducted on node deployment for sensor networks. Meguerdichian *et al.* [30] discuss several coverage problems in sensor networks. Their algorithms can identify a path that is least covered. Deploying more sensor nodes along this path can improve the sensor coverage. Howard *et al.* [20] present a deployment algorithm that incrementally deploys nodes based upon information gathered by previously deployed nodes. Zou *et al.* [50] use virtual forces to determine the positions to redeploy or relocate sensor nodes for better coverage. Poduri *et al.* [35] assume all sensor nodes are mobile and maintain a certain number of neighbors for each node. Cloqueur *et al.* [9] present a strategy that determines the number of sensors deployed in each step until the desired coverage is achieved. Wang *et al.* [44] design several algorithms for mobile sensor nodes to move after an initial deployment to increase the coverage. Xing *et al.* [45] discuss several geographic routing algorithms for sensing covered networks. Sensor coverage in sensor network is different from the robot coverage discussed in this paper. Sensor coverage assumes sensors are mostly static and observe the environment, while robot coverage in this paper assumes the robots move and sweep the area by their sensors.

Our research is different from previous studies in the following ways: 1) this paper develops efficient deployment strategies for mobile robots. The robots cooperatively cover an unknown area; 2) our paper considers both energy and timing constraints. We build power models for mobile robots and manage their speeds to maximize the traveling distance; 3) the algorithm can reduce three types of overhead with a smaller fleet size; and 4) we model environments with obstacles and estimate the additional distances the robots need to travel for covering an area with obstacles. An empirical rule is established from simulation results. With this rule, the deployment algorithms can be extended to environments with obstacles.

III. Problem Definition

Deployment is a complex problem. For simplicity, we make the following assumptions in this paper.

1) All robots are the same; they have the same amount of initial energy $\mathcal{E}$.
2) Each robot is equipped with sensors. The sensing range is d from the robot's center; d is called the *sensing distance*. The sensed region is a square of $2d \times 2d = 4d^2$ from the robot's center. The area covered by one robot is the product of $2d$ and this robot's traveling distance. This is an approximation of the sensing regions of real sensors. For example, a sonar ring can sense an approximate region of a circle, and we can use an inscribed square of this circle to approximate the sensing region. The adoption of a square-sensing region is to simplify the analysis of scanline coverage, mentioned next.
3) The robots travel along scanlines to cover a 2-D area; the time and energy for changing directions are not considered. In Section V-A, we explain in more detail why we choose scanlines and neglect turnings.
4) Our algorithm adopts a divide-and-conquer strategy, and requires little communication. When the carrier unloads a group of robots, the tasks (areas) are assigned to individual robots through wireless communication. After the assignment, the robots do not communicate with each other or with the carrier.
5) The carrier travels at a much higher speed than the robots, so the carrier's traveling time is negligible. This assumption is justified because the carrier can be a truck, a helicopter, or even an aircraft driven by a human rescuer. The time to unload n robots at the same location is $u(n) = u_0 + c \times n$, where c is a positive constant. The item u_0 is the time to stop the carrier, even if no robot is unloaded.

A. Traveling Speed

The sensor on each robot has a finite and constant sensing range. Thus, a robot can cover a larger area if the robot can travel a longer distance. A power model of a mobile robot has been built in our previous work [32]. The power model $p(v)$ is a convex function of robot traveling speed v. We use v_m to represent the maximum speed of the robot. The most energy-efficient speed, v_o, is defined as the speed at which the robot consumes the least energy to travel a unit distance, or $v_o = \arg\min_v\{p(v)/v\}, 0 < v_o < v_m$. Moving at the speed v_o is the most energy-efficient, and the robot can travel the longest distance with the same amount of energy, hence covering the largest area. However, if we consider the timing constraints, this may not be true. We use the symbol τ to denote the deadline, and $\mathcal{E}$ for the initial energy. All robots share the same deadline. For example, the deadline is 24 hrs after an earthquake, regardless of the time when a robot is deployed. The speed-management problem is to determine the traveling speed at which each individual robot can travel the longest distance under the timing and energy constraints.

B. Robot Deployment

We consider only one carrier in this paper, but our method can be extended to multiple carriers. At time $t = 0$, the carrier starts moving to the first unloading site. The area has to be covered before the deadline $t = \tau$. The carrier unloads robots at k different locations. At the ith location $(1 \le i \le k)$, n_i robots are unloaded. The total number of robots used is $n = \sum_{i=1}^{k} n_i$. The robots that are unloaded at the same location belong to the same group. Let $r_{i,j}$ $(1 \le i \le k, 1 \le j \le n_i)$ be a robot that is unloaded at the ith location. The area covered by this robot is

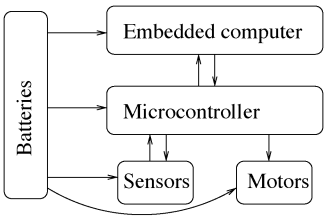


Fig. 1. Common component architecture of a robot.

denoted as $a_{i,j}$. The area covered by the ith group is $\sum_{j=1}^{n_i} a_{i,j}$. The total area covered by all robots is $\mathcal{A} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} a_{i,j}$. The deployment problem is to find a solution that minimizes the total number of robots n for covering a given area of size $\mathcal{A}$ under the energy and timing constraints.

C. Obstacles

The robots have no map of the area to be covered. However, we assume the density of obstacles is available. The obstacle density d_o is the ratio of area occupied by obstacles over the total area; it may be estimated by sampling a small portion of the total area or processing satellite images. In this paper, we estimate the additional distance each robot has to travel for detouring around obstacles. We use r_d as the ratio; in an open area, the ratio is one. Intuitively, r_d rises as d_o increases. When d_o is too high, however, a large portion of the area becomes inaccessible, and r_d may decrease.

IV. Power Models of Robots and Speed Management

Fig. 1 shows a common architecture for mobile robots. This architecture includes five major components: batteries; motors; sensors; a microcontroller; and an embedded computer. The most-often used energy sources are rechargeable batteries. The batteries need to be recharged after exhaustion. In some cases, such as rovers, solar-powered batteries are used. Motors, sensors, microcontrollers, and embedded computers are energy consumers. DC motors transform direct current into mechanical energy, and are used in robots as actuators. As the robots become more sophisticated, control, sensing, communication, and computation consume higher portions of energy. Robots use many kinds of sensors, such as encoders, vision, sonar, laser, and infrared rangers. The microcontroller handles low-level controls, such as directly controlling motors and polling readings from sensors. At the same time, it provide a programming interface for the embedded computer. The embedded computer has better computation ability, and is in charge of high-level controls, such as motion planning and coordination.

In the rest of this section, we will first model the power consumption of different components and then experimentally build power models for two different mobile robots. The first one is called PPRK, developed at Carnegie Mellon University (Pittsburgh, PA) [36]. The second robot is Pioneer 3DX by ActivMedia (Amherst, NH). Pioneer robots are popular in research communities [1], [16], [18].

A. Modeling of Power Consumption

1) Motion: Motors transform electrical energy into mechanical energy. The power consumption of the motors is the sum of the mechanical output power and the transforming loss. Let m be the robot's mass, and the ground friction constant be μ. When the robot travels with a speed of v and an acceleration of a, it needs a traction force of $m(a + g\mu)$. Therefore, the output mechanical power is $m(a + g\mu)v$, where g is the gravity constant. The motion power can be modeled as a function of the speed, the acceleration, and the mass:

$$p_m(m, v, a) = p_l + m(a + g\mu)v \tag{1}$$

where p_m is the motion power, and p_l is the transforming loss.

For DC motors, the power loss consists of armature loss, internal mechanical loss, and eddy-current loss. The power loss increases as the speed increases, but this relationship is nonlinear. For example, the eddy-current loss increases by the square of the speed [23]. The motion power efficiency can be defined as the inverse of the energy per unit distance, or $v/p(v)$. The efficiency will increase first as the speed increases, and then decrease due to the large power loss at a higher speed.

2) Sensing: Sensing power varies from different sensors and sensing frequencies. We can denote the sensing frequency by f_s. For video cameras, it is the number of frames per second; for laser rangers, it is the firing frequency. We conjecture that a linear function can model the power consumption of sensors

$$p_s(f_s) = c_{s_0} + c_{s_1} f_s \tag{2}$$

where p_s is the sensing power, and c_{s_0} and c_{s_1} are two positive constant coefficients. Their values depend on the sensors used. This model is validated by our experimental results.

3) Microcontroller and Embedded Computer: The microcontroller periodically sends commands to motors and sensors, polls sensors' readings, and communicates with the embedded computer. The microcontroller's tasks are usually fixed, and the power consumption of the microcontroller can be modeled by a constant.

The embedded computer is more complex than the microcontroller. Many studies have been devoted to simulation-based methods to estimate its power consumption [6], [26], [40]. The power consumption of the embedded computer may vary significantly for running different programs.

B. Power Model of PPRK

PPRK has three DC motors, three infrared sensors, and one microcontroller powered by four 1.2 V and one 9 V NiMH rechargeable batteries. The power consumption of the PPRK robot is divided into three parts: motion; sensing; and control. We use a data acquisition (DAQ) card from National Instrument Inc. (Austin, TX) to measure the current and the voltage. The DAQ is connected to a laptop computer so that the measurement facility can follow the robot while it is moving. Since the sensing frequency cannot be easily changed, the total power of sensing and control is measured constant at 0.998 W. We control the robot to move at constant speeds and measure the motion power (the total power of the three DC motors). The robot carries no additional load and moves on an indoor flat surface. The results are shown in Fig. 2. The power increases slowly before

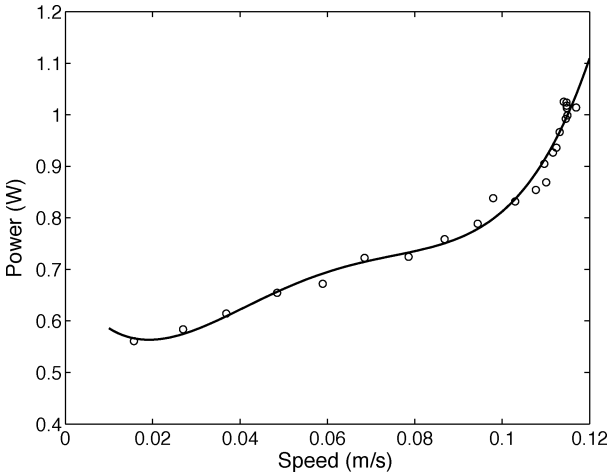


Fig. 2. Motion power at different speeds of PPRK.

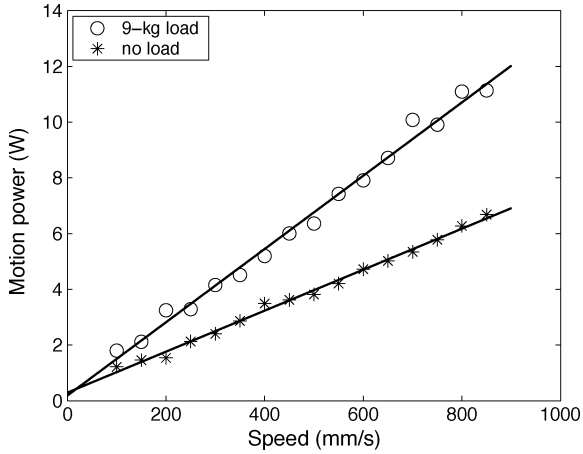


Fig. 3. Motion power at different speeds of Pioneer 3DX.

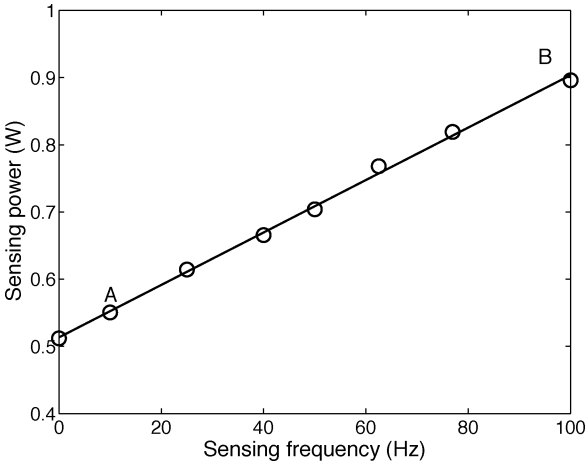


Fig. 4. Sonar sensors' power consumption.

reaching speed 0.1 m/s, but it increases super-linearly when the speed is larger than 0.1 m/s. The maximum speed of the PPRK robot v_m is 0.117 m/s, and the maximum power is 1.025 W. The optimal speed is $v_o = 0.110$ m/s, at which the power is 0.869 W. Including the sensing and control power, the energy efficiency at speed v_o is $0.11/(0.869+0.998) = 0.0589$ m/J. The energy efficiency at the maximum speed v_m is $0.117/(1.025+0.998) = 0.0579$ m/J or 1.9% lower than that at speed v_o. We fit the data by a fourth-degree polynomial, and the power model (including the sensing and control power) is

$$p(v) = 3.00 \times 10^4 v^4 - 6.97 \times 10^3 v^3 + 5.54 \times 10^2 v^2 - 1.44 \times 10^1 v + 1.68. \quad (3)$$

C. Power Model of Pioneer 3DX

The Pioneer 3DX weighs about 9 kg, and can carry at most a 22.5-kg load. The robot is powered by a 12 V lead-acid rechargeable battery. The robot has two DC motors driving two wheels. The DC motors are assembled with encoders. The robot has two arrays of sonar sensors, one in the front and one in the rear. Each array has eight transducers. A Hitachi H8S-based microcontroller is used to control motors and sensors, and it communicates with an embedded computer through a serial port. The microcontroller has many peripheral chips to support sonars, motors, joysticks, grippers, and bumpers. It is managed by a real-time operating system called AROS. We use a laptop computer to communicate with the microcontroller. The computer is a DELL PP04S with Pentium M 1.2 GHZ CPU and 256 MB memory. It runs the Windows XP operating system.

We measure the motion power of the robot when the robot moves straight at constant speeds. We also change the load of the robot. Fig. 3 shows the motion power of the robot runs at different speeds. In this figure, the lower set of data and the fitting line are for the robot without load; the upper set of data is for the robot with a 9-kg load. The motion power increases linearly as the speed increases. This figure does not show the super-linear increase at a higher speed, because in our experiments, speeds are lower than the maximum speed from the robot's specifications. The specified maximum speed can not be achieved in our experiments because the robot limits the motors' speed. The two motion power models (without load and with 9-kg load, respectively) are:

$$p_m(v) = 0.29 + 7.4v \quad (4)$$

$$p_m(v) = 0.19 + 13.1v. \quad (5)$$

Fig. 4 shows the total power consumption of the two sonar arrays at different sensing frequencies. The sonar sensing power model is

$$p_s(f_s) = 0.51 + 0.0039\ f_s. \quad (6)$$

The static power is 0.51 W, 76.9% of the total sensing power when the sensing frequency is 40 Hz. The power consumption increases as the sensing frequency increases; the power consumption at 10 Hz (point A) is 38.2% lower than that at 100 Hz (point B).

The power consumption of the microcontroller is very stable at 4.6 W (including power consumption of the Hitachi H8S and some peripheral control circuits) from our measurements. The power consumption of the embedded computer is estimated in the range of 8–15 W. These values are estimated by dividing the battery capacity by the time the computer can run with a fully charged battery when running different programs.

We assume the robot uses a fixed sensing frequency (25 Hz), the power of the microcontroller is constant at 4.6 W, and the power of the embedded computer is constant at 12 W. The robot moves without a load. The power of this robot is $0.51+0.0039\times 25+4.6+12+0.29+7.4v$. Therefore, the power model of this robot is

$$p(v) = 17.49 + 7.4v. \tag{7}$$

Since the power is a linear function of the speed, the maximum energy-efficient speed v_o is equal to the maximum speed v_m.

D. Speed Management

Speed management determines a robot's speed as a function of time. The purpose of speed management is to prolong the traveling distance under timing and energy constraints. Let $v(t)$ be a robot's speed at time t; the traveled distance is $\int_0^\tau v(t)dt$. Speed management finds $v(t)$ such that $v(t) = \arg\max_{v(t)}\{\int_0^\tau v(t)dt\}$. The solution has to satisfy three constraints: 1) $0 \le t \le \tau$; 2) $\int_0^\tau p(v(t)) \le \mathcal{E}$; and 3) $0 < v(t) \le v_m$. To decide the speed function $v(t)$, we need to understand the properties of power function $p(v)$. The robot power $p(v)$ is a monotonically increasing function of the speed, since the robot consumes more power at a higher speed. In general, it is a convex function of the speed $(d^2p(v)/dv^2) > 0$ [10], [32]. Notice that the power model of the Pioneer robot is a linear function. This is because the maximum speed in the experiments has not reached the point where the energy loss is comparable with the output mechanical power.

We prove that each robot should use a constant speed; the speed is determined by the energy capacity of the battery and the remaining time before the deadline.

Lemma 1: If the $\mathcal{E} \le \tau p(v_o)$, the robot can travel the longest distance at the most energy-efficient speed v_o.

Proof: We use $e(t)$ to represent the remaining energy at time t, and $e(0) = \mathcal{E}$. Also notice that the function $e(t)$ is nonnegative. The power is the derivative of consumed energy in terms of time; therefore, $de(t) = -p(v(t))dt$. The distance $\int_0^\tau v(t)dt = \int_0^\tau (v(t)/p(v(t)))p(v(t))dt = -\int_0^\tau (v(t)/p(v(t)))de(t) = \int_\tau^0 (v(t)/p(v(t)))de(t)$. Since the definition of v_o is

$$\arg\min_v (p(v)/v)$$

the distance $\int_\tau^0 (v(t)/p(v(t)))de(t) \le (p(v_o)/v_o)\int_\tau^0 de(t) = (p(v_o)/v_o)(e(0) - e(\tau)) \le (p(v_o)/v_o)\mathcal{E}$. The given condition of this lemma $\mathcal{E} \le \tau p(v_o)$ means that the robot can travel at speed v_o and run out energy before, or at least at, the deadline; in other words, the energy constraint is tighter than the timing constraint. Under this condition, the equalities are valid when the robot travels at the constant speed v_o, and the maximum distance is $\mathcal{E}v_o/p(v_o)$. ◇

Lemma 2: If $\mathcal{E} \ge \tau p(v_m)$, the robot can travel the longest distance at the maximum speed v_m.

Proof: The given condition means that the robot can travel at the maximum speed v_m and still have energy left at the deadline; namely, the timing constraint is tighter. Since the robot traveling at the highest speed can travel the longest distance $\tau \times v_m$, this lemma is obviously true. ◇

The first two lemmas are intuitive and determine the robot's speed for two extreme cases. When the energy is between the two extreme cases, the solution is not so obvious. The following lemma indicates that the robot should use a constant speed between v_o and v_m.

Lemma 3: If $\tau p(v_o) \le \mathcal{E} \le \tau p(v_m)$, there exists a speed v_1 $(v_o \le v_1 \le v_m)$ such that the robot can travel the longest distance and exhaust the energy at the deadline. The value of v_1 satisfies $p(v_1) = \mathcal{E}/\tau$.

Proof: Since $\mathcal{E} \le \tau p(v_m)$, the robot should exhaust the energy at the deadline; otherwise, the remaining energy is wasted. If $e(\tau) \ne 0$, the robot may increase the speed to consume more energy and travel farther. Therefore, we should achieve $e(\tau) = 0$ and $\int_0^\tau p(v(t)) = \mathcal{E}$. The power function $p(v)$ is a convex function of the speed v. By Jensen's inequality [14], $\overline{p(v)} \ge p(\overline{v})$, where the above bar means average. The average speed is $\overline{v(t)} = (1/\tau)\int_o^\tau v(t)dt$, and the average value of power is $\overline{p(v(t))} = (1/\tau)\int_o^\tau p(v(t))dt = \mathcal{E}/\tau$. Therefore, $p((1/\tau)\int_o^\tau v(t)dt) \le (\mathcal{E}/\tau)$.

Under the condition $\tau p(v_o) \le \mathcal{E} \le \tau p(v_m)$, there exists a speed v_1 $(v_o \le v_1 \le v_m)$ satisfying equation $p(v_1) = \mathcal{E}/\tau$. This can be proved by the monotonic increasing property of the power function $p(v)$. Continuing the last formula in the last paragraph, we have $p((1/\tau)\int_o^\tau v(t)dt) \le p(v_1)$. Also by the monotonic property, $(1/\tau)\int_o^\tau v(t)dt \le v_1$. Therefore, the traveling distance $\int_o^\tau v(t)dt \le v_1\tau$. The equality holds when $v(t) = v_1, 0 \le t \le \tau$. This shows that if the robot travels at the constant speed v_1, it can travel the longest distance. In this case, the robot will exhaust the energy exactly at the deadline. ◇

The following theorem concludes our speed-management strategy.

Theorem 1: With an energy constraint $\mathcal{E}$ and a timing constraint τ, a robot can maximize the distance it can travel with a constant traveling speed. This constant speed is determined by the following rules:

- if $\mathcal{E} \le \tau p(v_o)$, the speed is v_o;
- if $\mathcal{E} \ge \tau p(v_m)$, the speed is v_m;
- if $\tau p(v_o) \le \mathcal{E} \le \tau p(v_m)$, the speed is v_1 $(v_o \le v_1 \le v_m)$, at which the power $p(v_1)$ is equal to $\mathcal{E}/\tau$.

V. Deployment Strategy

This section describes our deployment strategy. We first explain three types of overhead during deployment. Our method calculates the area covered by each group of robots; then it determines the number of groups.

A. Overhead in Deployment

This paper focuses on deployment, and the coverage algorithm is to help us understand different deployment strategies. There are several ways to cover an area, as discussed in our previous paper [32]. This paper considers scanlines only, although in some situations, some other paths like spiral curves are more efficient in coverage. Scanline coverage is simple, and also used

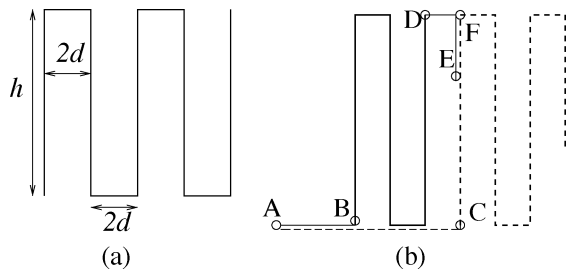


Fig. 5. (a) Scanline-covering route. (b) Three robots are unloaded at A. The starting locations are A, B, and C. The segments $\overline{AB}$ and $\overline{AC}$ represent the dispersing overhead. The second robot runs out of energy and stops at E.

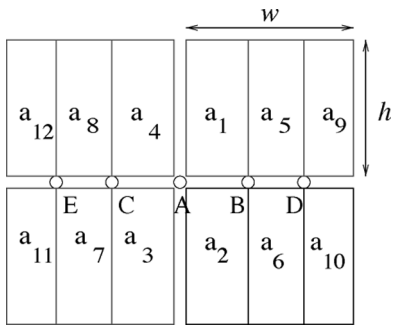


Fig. 6. Area is covered by a group of 12 robots. The areas with subscripts from 1 to 12 are the areas covered by these robots. The areas are symmetric to the unloading location A.

in some other studies [24], [27]. We neglect turnings to simplify our following analysis. This is a reasonable simplification when the scanlines are long, as shown in [32]. Fig. 5(a) shows a scanline route of one robot. The height is h; the distance between two adjacent lines is $2d$ because the sensing distance is d from the robot's center. There are three types of overhead that increases the number of robots needed to cover an area.

1) The first type of overhead is the time spent for unloading the robots. Because unloading takes time, a robot that is unloaded later has less time before the deadline.
2) The second type of overhead comes from the time and the energy spent by each robot to reach its own working area after being unloaded. Because robots are unloaded in groups, they need to disperse from the unloading location to their individual working areas.
3) The third type of overhead occurs when a robot cannot finish a scanline due to energy or timing constraints, or both. As a result, another robot has to cover the rest of this line. We call this *fragmentation overhead*, because it is similar to sector fragmentation of hard disks. As h increases, fragmentation overhead increases.

They are called unloading, dispersing, and fragmentation overhead in the rest of this paper. These types of overhead are related. Fig. 5(b) illustrates the second and third types of overhead. Suppose three robots are unloaded at location A. Their starting locations are A, B, and C, respectively. The first robot spends no dispersing time. The second robot's dispersing overhead is to travel through $\overline{AB}$; the third robot's dispersing overhead is the distance $\overline{AC}$. Suppose the second robot stops at E when it exhausts its energy after completing three scanlines, shown in solid lines. The third robot has to cover $\overline{CE}$; otherwise, this area is not covered by any robot. To simplify our analysis, each robot finishes only integer numbers of scanlines. In other words, the second robot stops at D because its remaining energy and time do not allow the robot to finish another scanline, and the third robot covers $\overline{CF}$.

B. The Area Covered by One Group

Our method first considers the dispersing and fragmentation overhead of a single group. To reduce the dispersing overhead, the robots' starting locations should be close to the unloading location. Since the robots travel along scanlines and cover rectangles, our method covers the four quadrants symmetrically in a 2-D Cartesian coordinates centered from the unloading location. Fig. 6 shows an example of an area covered by a group of 12 robots. In the figure, point A is the unloading location of the whole group and the starting point of the first four robots. The first four robots move in different directions from point A to cover a_1, a_2, a_3, and a_4. Point B is the starting location of the fifth and sixth robots. Point C is the starting location of next two robots, the seventh and eighth. Because the fifth robot spends time traveling across $\overline{AB}$, its covered area cannot be larger than a_1, i.e., $a_1 \geq a_5$. Because these four quadrants are symmetric, we can obtain the relationship $a_1 = a_2 = a_3 = a_4 \geq a_5 = a_6 = a_7 = a_8 \geq a_9 = a_{10} = a_{11} = a_{12}$. In the rest of this paper, we consider the first quadrant only.

Let w be the width of the area covered by one quarter of the robots in the same group, as shown in Fig. 6. The total dispersing distance of the three robots covering a_1, a_5, and a_9 is $0+\overline{AB}+\overline{AD} \approx 0+(w/3)+(2w/3) = w$. We can extend this observation to more robots. If this group contains 4ψ robots (ψ for each quadrant), the total dispersing distance in one quadrant is approximately $(w/\psi)+(2w/\psi)+\ldots+((\psi-1)w/\psi) = (\psi-1)w/2$. The average dispersing distance for each robot is $((\psi-1)w/2)(1/\psi)$ converges to $w/2$ for many robots. Meanwhile, the average fragmentation overhead for each robot is $h/2$. Using these two values of average overhead, our strategy chooses values of w and h so that they are as close as possible. The rationale is explained below.

All robots in the same group can travel the same maximum distance, because each has the same amount of energy and time. Let this distance be l. The available time before the deadline is τ. Suppose the robots travel at speed v and consume power p. The traveling time of a robot is at most $\mathcal{E}/p$, and hence, the robot can operate at most $\min(\mathcal{E}/p, \tau)$ time, and travel at most $v \times \min(\mathcal{E}/p, \tau)$ distance. This is the value of l. Each robot can sense a region of $2d$ from its center; hence, to cover the area wh, the total traveled distance of the whole group is $wh/2d$. This area is reduced due to the overhead, so $\psi l \approx (wh/2d) + \psi((w/2)+(h/2))$. Using the Lagrange multiplier method to maximize the total covered area, we can obtain the condition $w = h$. By setting $w = h$, we obtain the relationship $\psi l \approx (h^2/2d)+\psi h$. With the value of ψ, we can determine the value of h, i.e., the height of the scanlines. The area covered by this group is $4wh$ (four quadrant). We will calculate the number of robots in the group ψ in the next subsection.

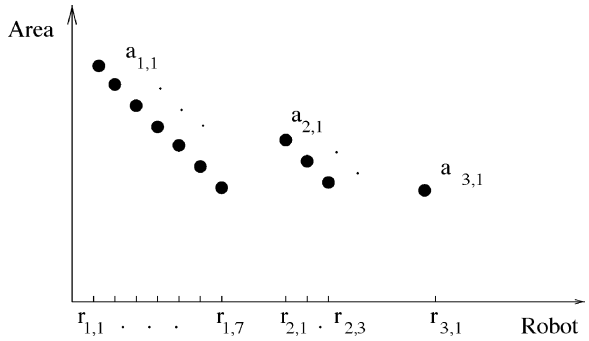


Fig. 7. Deployment with three groups.

C. Number of Groups and Group Sizes

Minimizing the number of robots used is equivalent to maximizing the average area covered by each robot. The area covered by the robots in each group can be ordered by the coverage sizes. When more robots are unloaded at the same location, the minimum size of this group decreases for two reasons. First, it takes longer to unload the whole group. Second, some robots need to travel farther to reach their starting locations. Our method adopts the following rules to enlarge the average area covered by each robot.

1) The minimum area covered by the robots in each group should be close. Let a_{i,n_i} and a_{j,n_j} be the minimum areas covered by the ith and jth groups. If a_{i,n_i} is much smaller than a_{j,n_j} due to the dispersing overhead, then we should use fewer robots in the ith group and more robots in the jth group. By adjusting the group sizes, we can enlarge the *average* area covered by the robots in both groups.
2) An earlier deployed group should have a group size larger than or equal to those of latter deployed groups. According to the first rule, they all have similar minimum areas. However, later deployed groups have less time before the deadline.
3) For two deployments that satisfy the above two rules and can cover the same assigned area, the one that has a smaller size of the first group is better because it makes the minimum area larger.

These three rules together determine the number of groups and the sizes of groups. Fig. 7 shows an example deployment with three groups of seven, three, and one robot(s), respectively. The group sizes are in decreasing order. The vertical axis is the area covered by individual robots. The maximum areas of the three groups are $a_{1,1}$, $a_{2,1}$, and $a_{3,1}$; they have a relationship of $a_{1,1} \geq a_{2,1} \geq a_{3,1}$, because later groups have less time before the deadline. The minimum areas of the three groups are $a_{1,7}$, $a_{2,3}$, and $a_{3,1}$; they are close to each other to increase the average area.

The next section presents our algorithm generating deployment solutions that satisfy the above three rules.

D. Deployment Algorithm

ROBOT-DEPLOYMENT$(\mathcal{E}, \tau, \mathcal{A})$

1 $\mathcal{E}_t \leftarrow \mathcal{E}, \tau_t \leftarrow \tau, \mathcal{A}_t \leftarrow \mathcal{A}$
  /* $\mathcal{E}$: energy, τ: deadline, $\mathcal{A}$: total area to cover */
  /* $\mathcal{E}_t$, τ_t and $\mathcal{A}_t$: used to remember initial values */
2 $n_1 \leftarrow 1$
3 $n_p \leftarrow n_1$
4 $\tau \leftarrow \tau - u(n_p)$
5 $\mathcal{A} \leftarrow \mathcal{A}$–GET-TOTAL-AREA$(\mathcal{E}, \tau, n_p)$
6 $min_area \leftarrow$GET-MINIMUM-AREA$(\mathcal{E}, \tau, n_p)$
7 $flag \leftarrow 0$ /* no solution yet */
8 **while** $flag = 0$
9 **do** $n \leftarrow n_p$ /*n is the fleet size */
10 $diff \leftarrow \infty$ /* initialization */
11 **while** $\mathcal{A} > 0$ **and** $\tau > 0$
12 **do for** $i \leftarrow 1$ **to** n_p /* select next group size */
13 **do** $\tau \leftarrow \tau - u(i)$
14 $temp \leftarrow$GET-MINIMUM-AREA$(\mathcal{E}, \tau, i)$
15 **if** $i = 1$ **and** $temp < min\_area$
16 **then** $\tau \leftarrow -1$, $\mathcal{A} \leftarrow 1$
17 **break**
18 /* find the closest minimum area */
19 **if** $|min\_area - temp| < diff$
20 **then** $diff \leftarrow |min\_area - temp|$
21 $x \leftarrow i$
22 $\tau \leftarrow \tau + u(i)$
23 **if** $\tau < 0$
24 **then break**
25 $\tau \leftarrow \tau - u(x)$
26 $\mathcal{A} \leftarrow \mathcal{A}$–GET-TOTAL-AREA$(\mathcal{E}, \tau, x)$
27 $n \leftarrow n + x$
28 $n_p \leftarrow x$
29 $min_area \leftarrow$GET-MINIMUM-AREA $(\mathcal{E}, \tau, x)$
30 **if** $\tau > 0$ /* before the deadline */
31 **then** $flag \leftarrow 1$
32 **else** $n_1 \leftarrow n_1 + 1$ /* change first group size */
33 $\mathcal{E} \leftarrow \mathcal{E}_t, \tau \leftarrow \tau_t, \mathcal{A} \leftarrow \mathcal{A}_t$
34 $n_p \leftarrow n_1$

```
35              τ ← τ − u(n_p)
36              A ← A−GET-TOTAL-AREA(E, τ, n_p)
37              min_area ←GET-MINIMUM-AREA
                  (E, τ, n_p)
    /* adjust the size of last group */
38  n ← n − x
39  A ← A+GET-TOTAL-AREA(E, τ, x)
40  τ ← τ + u(x)
41  for j ← 1 to x
42     do u ← τ − t_u(j)
43        if A ≤GET-TOTAL-AREA(E, τ, j)
44          then n ← n + j
45               break
46          else τ ← τ + u(j)
47  return n
```

Our algorithm is called SPACA. It sets the size of the first group, and then determines the sizes of the other groups using a greedy approach. Each group's size depends on only the sizes of the previous groups. The algorithm starts from a small size of the first group, then increases the size until a solution is found. The size of the first group is initialized to one. This corresponds to the third rule in the last section. The outer **while** loop finds the size for the first group until a solution is found.

The inner **while** loop assigns the sizes of the other groups until either the area $\mathcal{A}$ has been covered or the deadline has passed. The variable n_p keeps the size of the latest assigned group. The minimum area covered by the previous group, i.e., a_{n_p}, is calculated, and it is recorded by the variable min_area. According to rule 2), the next group size is at most n_p. In the **for** loop from line 12, the algorithm computes the minimum areas of the next groups with sizes from 1 to n_p, and selects the size when the next group has the closest minimum area with min_area. This is required by the first rule. The functions GET-TOTAL-AREA and GET-MINIMUM-AREA compute the total area and the minimum area covered by one group of robots, respectively.

There are three possible cases making the algorithm leave the inner **while** loop. The first case happens in the first **if** statement inside the inner **while** loop at line 15. When the next group size is 1 and the minimum area is less than the previous minimum area, the first rule is impossible to fulfill. Therefore, we assign a negative value to τ and leave the inner **while** loop. In the second case, the remaining time τ is nonpositive, while the remaining area $\mathcal{A}$ is still positive. This means the area has not been fully covered but no time is left. The third case happens when the area is covered and there is still time. In the first two cases, the algorithm does not find a solution. It increases n_1 by 1, renews the parameters, and continues the outer **while** loop. In the third case, a successful deployment is found and the variable $flag$ is set to leave the outer **while** loop. Because the determination of the last group size depends only on the comparison of minimum areas, not the remaining area, we may use more robots than necessary in the last group. The last steps adjust the size of the last group based on the remaining area. This algorithm will report the situation when it is impossible to cover the area due to a short deadline.

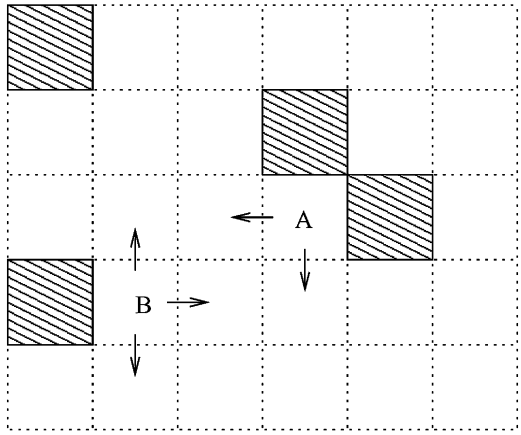


Fig. 8. Grid lines, cells, and neighbor cells.

The complexity of this deployment algorithm is $O(n^3)$, where n is the total number of robots deployed. This is analyzed as follows. The time complexity of GET-MINIMUM-AREA$(\mathcal{E}, \tau, x)$ or GET-TOTAL-AREA$(\mathcal{E}, \tau, x)$ is linear to the number of robots, i.e., $O(x)$. The critical statement is in line 14 for analyzing the complexity of this algorithm. For each specific i, the complexity of line 14 is $O(i)$. Furthermore, the complexity of the for loop, from line 12 to line 22, at a specific group size n_p, is $O(1)+O(2)+\ldots+O(n_p) = O(n_p^2)$. For a specific n_1, assuming the algorithm tries through k groups with sizes of $n_1, n_2, \ldots, n_k$, the complexity is $O(n_1^2)+O(n_2^2)+\ldots+O(n_k^2) = O((n_1 + n_2 + \ldots + n_k)^2) = O(n^2)$, because $n_1 + n_2 + \ldots + n_k \leq n$. The value of n_1 can be increased from 1 to n, and for each n_1, the complexity is $O(n^2)$. Therefore, the complexity of this deployment algorithm is $n \times O(n^2) = O(n^3)$.

VI. Environments With Obstacles

In this section, we present two probabilistic models for obstacles. An empirical rule is developed to estimate the additional distance robots have to travel for detouring around obstacles, and calculate the distance ratio r_d for different obstacle density d_o. We use simulation to validate the rule.

A. Obstacle Model

To model the environments, we make the following assumptions: 1) the area is divided into equal-size square cells; 2) each cell is either occupied by an obstacle or free; 3) obstacles are static and do not move; 4) a robot can move from one cell to its four neighbors: upper; down; left; and right; and 5) the density of the obstacles is known in advance, but the obstacles' exact locations are unavailable. Fig. 8 shows an area with 5-row×6-column cells. Four cells with shading are obstacles; the obstacle density is $(4/30) \approx 13.3\%$. A robot at cell "A" can move only

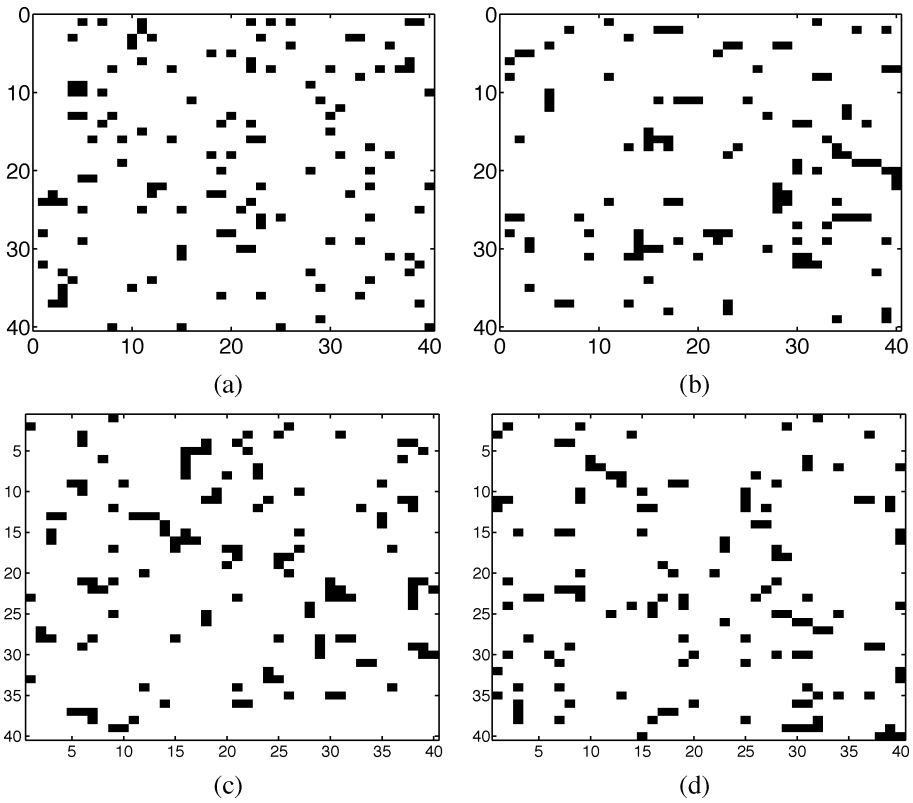


Fig. 9. Four examples of environments filled with probabilistic obstacles with obstacle density 8%. (a) Generated by random model. (b), (c), and (d) Generated by clustering model with different (k_1, k_2).

to its left and down cells; at point "B," the robot can move to upper, right, and down cells.

We develop two environment models: random model and clustering model. In the random model, the probability of one cell occupied by an obstacle is independent of its neighbors and is equal to d_o. In the clustering model, the obstacle possibility of one cell depends on its upper and left neighbors. We use four values p_{ul}, p_u, p_l, and p_n to represent obstacle probabilities of a cell in four different situations: 1) when both its upper and left neighbors are obstacles; 2) only the upper cell is an obstacle; 3) only the left cell is an obstacle; and 4) neither neighbor is an obstacle. Since an environment generated by the clustering model has an obstacle density of d_o, the following equation must hold: $d_o^2 p_{ul} + d_o(1-d_o)(p_l + p_u) + (1-d_o)^2 p_n = d_o$. We further assume that $p_u = p_l = k_1 p_{ul}$ and $p_n = k_2 p_{ul}$, where k_1 and k_2 are two constants. The value of p_{ul} can be calculated as $p_{ul} = d_o/(d_o^2 + 2d_o(1-d_o)k_1 + (1-d_o)^2 k_2)$, and the four probabilities can be uniquely determined. The values of k_1 and k_2 model the likelihood that a cell is occupied if its neighbors are occupied. When $k_1 < 1$ and $k_2 < 1$, p_{ul} is larger so a cell is more likely to be occupied if its upper and left neighbors are occupied. When $k_1 = k_2 = 1$, $p_{ul} = p_u = p_l = p_n = d_o$ and the clustering model degenerates to the random model. Fig. 9 shows four areas generated with $d_o = 8\%$. In Fig. 9(b)–(d), the values of (k_1, k_2) are (0.6, 0.1), (0.8, 0.2), and (0.7, 0.3), respectively.

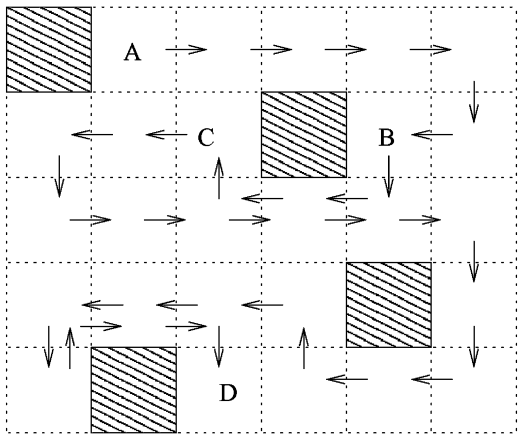


Fig. 10. Scanline covering an environment with obstacles.

B. Distance Ratio

Fig. 10 shows an example of scanning an area of 30 cells with $d_o = (4/30) \approx 10.33\%$. This example illustrates how we count the distance to cover an area with obstacles. Since the top left cell is an obstacle cell, the robot starts from cell "A." The robot moves to the right, until the end of the first row. Then, it moves down to the right end of the second row and heads left. At cell "B," it encounters an obstacle cell. It observes that the next unvisited cell is "C." From "B" to "C," there are two shortest

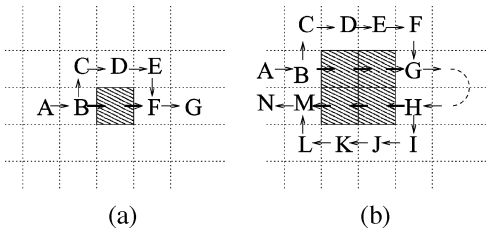


Fig. 11. Obstacles and traveling steps.

paths, one from above, the other from below. Suppose the robot chooses the lower one, and reaches cell "C." It continues moving toward the left until it finishes the second row. The robot then moves down to the left side of the third row and heads right. The middle three cells of the third row are visited by the robot twice. At the fourth row, the robot encounters an obstacle, and chooses the lower shortest path. At the last row, the robot encounters an obstacle cell again, and it chooses the only shortest path, moving upward to reach the next accessible cell "D." The left three cells of the fourth row are visited twice. The right three cells of the last row have already been visited, so the robot stops at cell "D." The arrows in the figure show the whole path from "A" to "D." There are a total of four obstacle cells, and the robot visits six cells twice to avoid the obstacles. The length of this path is $5 \times 6 + 6 - 4 = 32$ steps. Without the obstacles, the robot scans the whole area using $5 \times 6 = 30$ steps (moving from one cell to one of its neighbors is counted as one step). The distance ratio is $r_d = (32/30) \approx 1.067$.

To derive the relationship between r_d and d_o, we analyze two special cases as examples shown in Fig. 11. In Fig. 11(a), a robot detours around an isolated obstacle in the sequence shown by the letters and the arrows. It takes two more steps to avoid this obstacle. In Fig. 11(b), a robot detours around four obstacles clustered as a square. The letters and arrows show the traveling sequence. At cell "G," the robot travels right until reaching the right boundary, turns down to the next row, and then travels left to cell "H." The steps from cell "G" to cell "H" are omitted in the figure by the dash curve. It takes four extra steps to avoid the four obstacles. Suppose an area contains a total of N cells. If there is no obstacle, it takes N steps to cover this area. One obstacle cell increases d_o by $1/N$, and one extra step increases r_d also by $1/N$. If all the obstacles are isolated, similar to the obstacle in Fig. 11(a), one obstacle cell will incur two extra steps. Therefore, we can model the relationship as $r_d = 1 + 2 \times d_o$. This analysis can be applied to the clustering obstacles, as well. If all the obstacles are clustered as in Fig. 11(b), then the relationship is $r_d = 1 + d_o$. Therefore, we model the relationship between r_d and d_o by the following equation, where α is a positive constant:

$$r_d = 1 + \alpha \times d_o. \tag{8}$$

When the obstacle density d_o is large, the obstacle-clustering shapes are more complex. In the following paragraphs, we use simulations to decide the appropriate value of α, and also the applicable range of d_o.

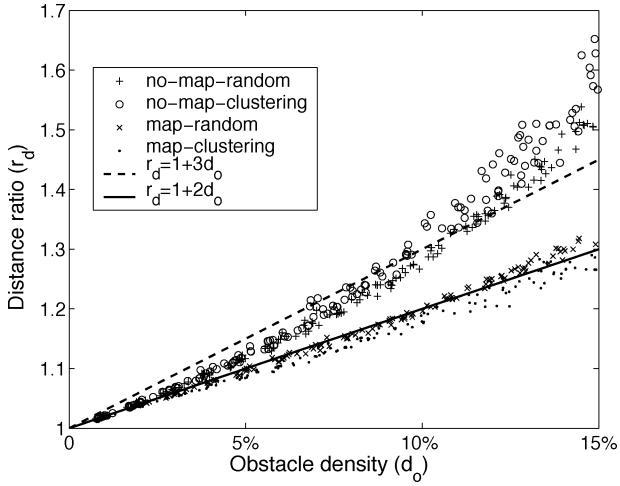


Fig. 12. Distance ratio versus obstacle density.

To validate this relationship, we perform extensive simulations of different maps with different values of d_o. The simulator first randomly generates an area using either model, and then simulates the scanline-covering process. Two sets of simulations are performed, and both of them assume the robot knows its current location. The first set assumes that the robot has the map of the area, obstacles, and its own location. The robot scans the area without considering timing or energy constraints. The robot starts from the top left toward the right, following the scanlines until all accessible cells have been visited. If the robot encounters obstacles, it picks the next unvisited accessible cell along the scanlines, and then chooses a shortest path from the current cell to that unvisited cell. The robot remembers all the cells it has visited. This simulation provides a lower bound for r_d, because the robot has the map and can determine the next unvisited accessible cell.

In the second set of simulations, the robot has no map. The robot starts from the top left toward the right, following the scanlines until covering the last row of cells. When the robot encounters obstacles, it detours around obstacles until the next cell along the scanline is reached. There are two detouring directions: clockwise and counterclockwise. When the robot moves toward the right, it detours in the counterclockwise direction first to the uncovered cells. In this direction, it can avoid the cells that have been covered. Since the robot has no map, it does not know whether this is a viable detouring. If it fails, the robot detours around the obstacles in a clockwise direction. When the robot moves toward the left, it tries the clockwise direction first. This kind of simulation adopts a simple, yet effective, covering algorithm. The robot travels a longer distance than the result in the first set of simulations because the robot has no map.

We run simulations with obstacle densities ranging from 1% to 15%. Fig. 12 shows the simulation results. The top two sets of data ($+$ and $\circ$) are from the second set of simulations. They are called "no-map-random" and "no-map-clustering" for the two probabilistic models. The other two sets of data ($\times$ and $\cdot$) are from the first set of simulations. They are called "map-random" and "map-clustering." Two lines enclose the range between $r_d = 1 + 2 \times d_o$ and $r_d = 1 + 3 \times d_o$. Fig. 12 shows that with maps, the robot can travel shorter distances to cover

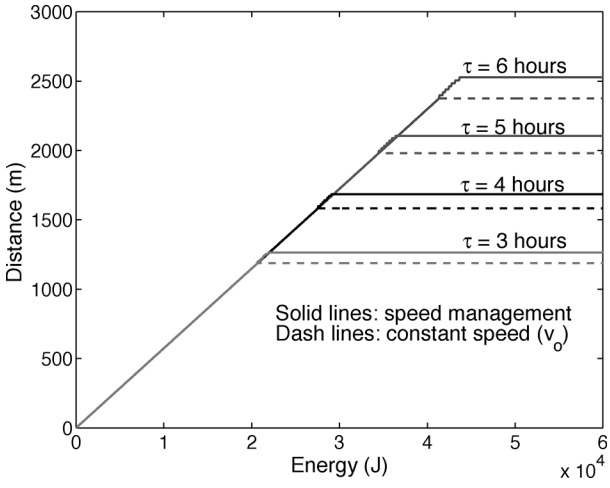


Fig. 13. Traveling distance with different energy, PPRK.

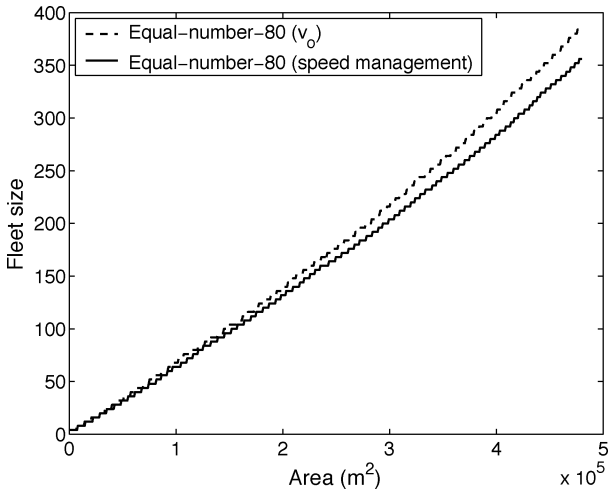


Fig. 14. Fleet size and speed management, PPRK.

the same area. The results from models are consistent. This shows that at a low d_o, the two models have negligible difference. When the density is less than 10%, the relationship $r_d = 1 + 3 \times d_o$ is valid for the no-map simulations. With maps and $d_o < 10\%$, the relationship can be closely approximated by $r_d = 1 + 2 \times d_o$. From our stimulation results, the value of α in (8) is between 2 and 3 for $d_o \leq 10\%$.

VII. Case Studies

In this section, we use the two robots PPRK and Pioneer 3DX to conduct case studies. Their power models have been developed in Sections IV-B and IV-C, respectively. The initial energy $\mathcal{E}$ of the two robots is estimated from the specifications of rechargeable batteries. PPRK uses four 1.2 V and one 9 V Ni-Cd batteries; Pioneer uses one 12 V lead-acid battery. The value of $\mathcal{E}$ for PPRK is 25 000 J, and the value for Pioneer is 311 040 J. We compare our method with two other solutions. The first is equal-number deployment by unloading the same number of robots each time until the assigned area is covered. We choose equal-number-40 (40 robots each time) and equal-number-80 (80 robots each time), because they represent relatively smaller and larger group sizes than our method. The second is one-unloading deployment by unloading all robots at one location. The unloading time is $u(n) = 600 + 2.5n$ s for n robots. The sensing distance used is 0.8 m.

A. Speed Management

Our speed-management method is to determine a speed between v_o and v_m that can maximize the traveling distance under energy and timing constraints. For the Pioneer robot, the speed is the maximum speed, since $v_o = v_m$.

Fig. 13 shows the difference between our speed-management method and the constant speed (v_o) method (the robots always travel at speed v_o). The timing constraint τ is 3, 4, 5, and 6 hrs, respectively; the energy $\mathcal{E}$ varies from 0 to 2.5×10^4 J. For $\tau = 3$ hrs, when $\mathcal{E}$ is less than 2.07×10^4 J, the two methods are the same. This is because our speed-management method chooses v_o when $\mathcal{E}$ is low. When $\mathcal{E}$ is higher, our method outperforms the constant speed (v_o) method by 6%. When $\tau = 4$, 5, and 6 hrs, our method also outperforms the constant-speed method when $\mathcal{E}$ is high. Fig. 14 shows the effect of speed management on fleet size ($\tau = 3$ hrs, $\mathcal{E} = 2.5 \times 10^4$ J). The fleet size is calculated using the deployment strategy of equal-number-80. When using speed management, we can use a smaller number of robots to cover the same amount of area. For example, we use $(304 - 284)/304 = 6.6\%$ fewer robots to cover 4×10^4 m^2 with speed management.

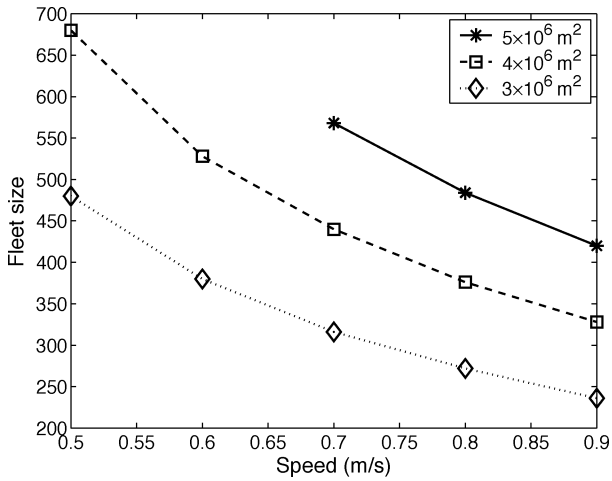


Fig. 15. Fleet size verses speed for Pioneer 3DX to cover different areas, time = 3 hrs, energy = 311 040 J.

Fig. 15 shows the fleet size of Pioneer 3DX robots to cover different areas with sizes of 4×10^6, 5×10^6, and 6×10^6 m^2, respectively. The robots travel at different speeds and cover the area within 3 hrs. We use our deployment algorithm SPACA. It is clearly seen that the fleet size decreases as the speed increases. This is because at a higher speed, the robots can travel farther before the deadline, and Pioneer's energy efficiency is higher. The figure also shows that at the same speed, a larger area requires a larger fleet size. Notice that when speed is lower than 0.7 m/s, SPACA can not cover 5×10^6 m^2 under the energy and timing constraints. For the rest of this paper, we assume our speed-management method is used.

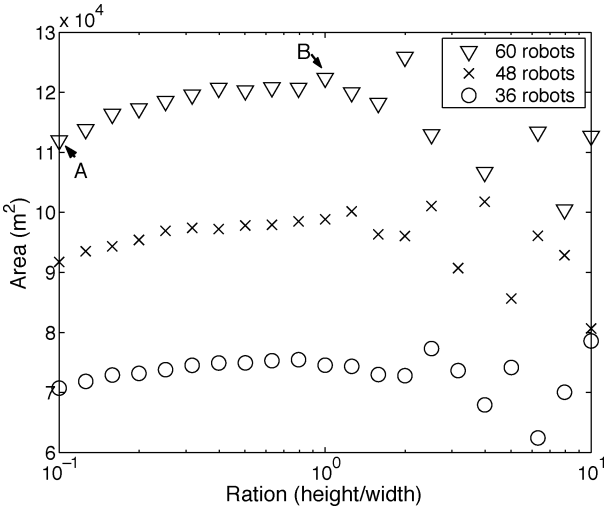


Fig. 16. Area covered by different number of robots with different ratios of height and width (6 hrs, 25 000 J, PPRK).

B. Robot Deployment in Open Areas

The height and width of the area covered by a group should be close, in order to reduce the dispersing and fragmentation overhead. Fig. 16 shows the size of the area covered by a different number of PPRK robots with different height–width ratios. The robots have 6 hrs before the deadline, and each of them has energy of 25 000 J. When the ratio is less than one, the covered area increases as the ratio increases. This is because the dispersing overhead dominates when the width is larger, and the overhead decreases as the ratio increases. Point A indicates the area when $h = 0.1w$ for 60 robots. Point B has $h = w$, and the covered area increased by more than 9%. When the ratio of h and w exceeds one, the covered area becomes unstable, sometimes increasing and sometimes decreasing. The reason is that fragmentation overhead is sensitive to the value of h. The figure shows three different group sizes. With 60 robots, an area larger than B can be covered when h is $4w$. However, the same ratio $h = 4w$ can cover a smaller area with 48 robots or 36 robots. Moreover, the variation is significant when $h \gg w$. Hence, we choose $h \approx w$ in our algorithm, because this provides stably large covered areas.

Figs. 17 and 18 show the number of needed robots for covering different area sizes using PPRK. Figs. 19 and 20 are simulation results of Pioneer 3DX. Since pioneer robots have higher speeds and carry more energy, they can cover more area than the same number of PPRK robots. Figs. 17 and 19 have a timing constraint (τ) of 3 hrs, while Figs. 18 and 20 have 6 hrs. For equal-number deployment, we choose two different numbers, 40 and 80. In all four figures, our method requires the fewest robots to cover the areas. To cover the same area, more robots are needed when the deadline is earlier (Figs. 17 and 19). In Fig. 17, equal-number-80 needs almost the same number of robots as our method. Equal-number-40 is better than one-unloading when the area is small, and one-unloading is better than equal-number-40 when the area is larger than 4.1×10^5 m^2. However, both equal-number-40 and one-unloading are inferior to our method. In Fig. 19, the two curves of

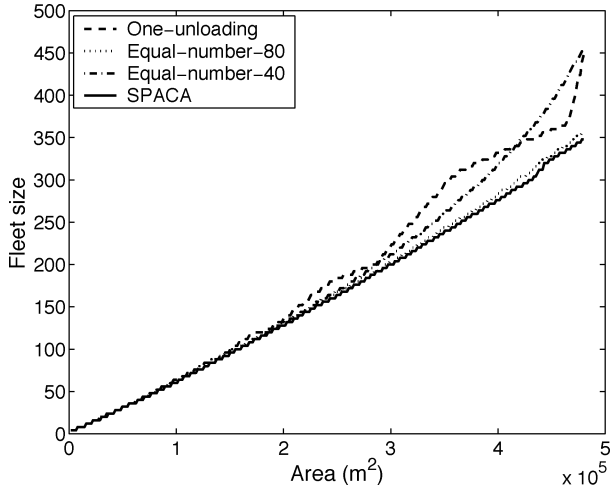


Fig. 17. Fleet size versus area for PPRK, time = 3 hrs, energy = 25 000 J.

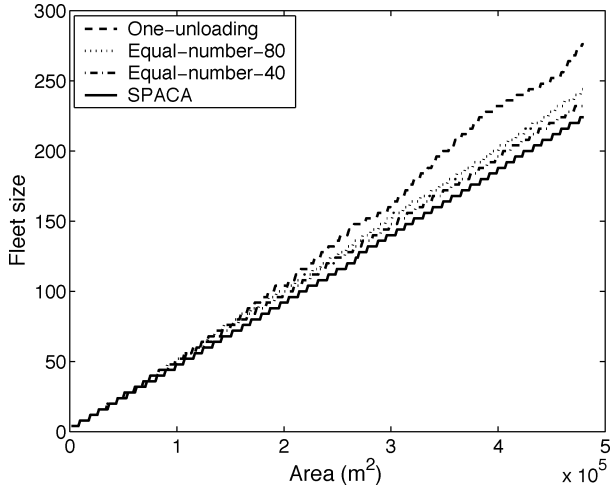


Fig. 18. Fleet size versus area for PPRK, time = 6 hrs, energy = 25 000 J.

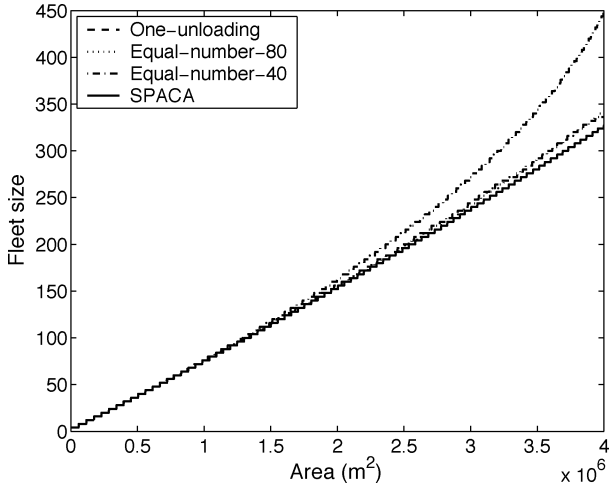


Fig. 19. Fleet size versus area for Pioneer 3DX, time = 3 hrs, energy = 311 040 J.

equal-number-80 and one-unloading are almost indistinguishable, and equal-number-40 requires the most robots. Figs. 18

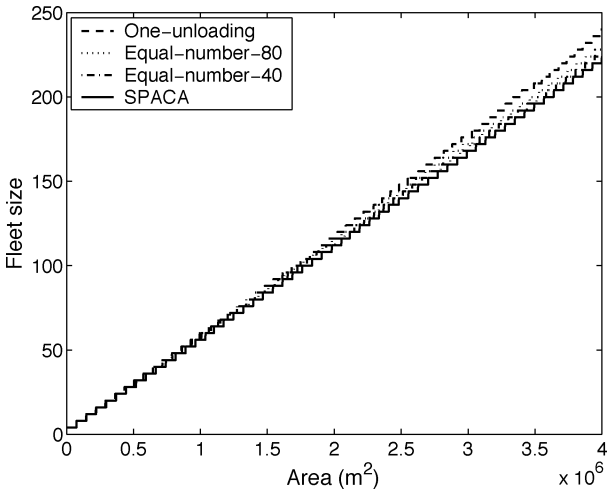


Fig. 20. Fleet size versus area for Pioneer 3DX, time = 6 hrs, energy = 311 040 J.

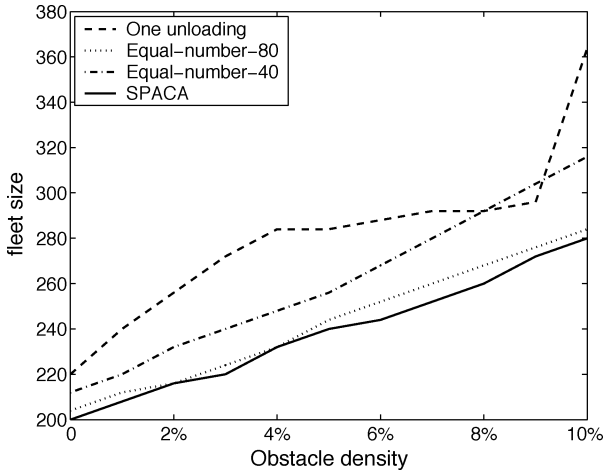


Fig. 21. Fleet size versus area for PPRK, time = 3 hrs, energy = 25 000 J, area 3×10^5 m^2.

TABLE I
DEPLOYMENT FOR COVERING 4×10^5 m^2 WITHIN 3 HRS WITH PPRK

Group number	1	2	3	4
Group size	140	84	52	4
Average area	1475	1422	1365	1397
Minimum area	1219	1206	1365	1397
Total area	206485	119444	70992	5588

TABLE II
DEPLOYMENT FOR COVERING 4×10^6 m^2 WITHIN 3 HRS WITH PIONEER

Group number	1	2
Group size	240	88
Average area	12410	11814
Minimum area	11657	11814
Total area	2978471	1039610

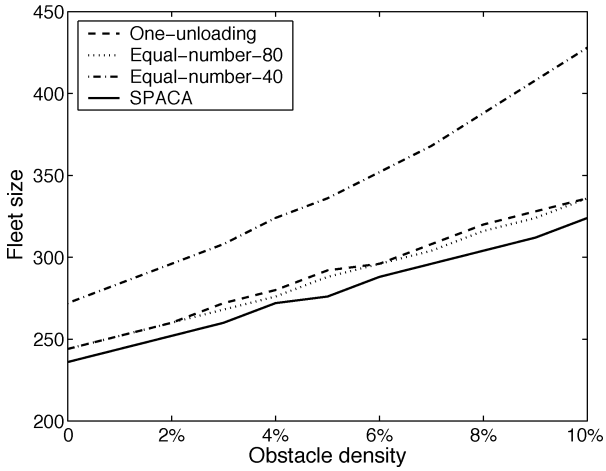


Fig. 22. Fleet size versus area for Pioneer 3DX, time = 3 hrs, energy = 311 040 J, area = 3×10^6 m^2.

and 20 allow a longer deadline, so unloading time is less important. In contrast, the dispersing and fragmentation overhead play more important roles. Deployments with a small group size can save dispersing and fragmentation overhead, thus requiring fewer robots than the deployments with a large group size. From both figures, we can observe that equal-number-40 is better than equal-number-80, and equal-number-80 is better than one-unloading.

Tables I and II show the details of two deployments generated by our method. Table I uses a total of 280 PPRK robots to cover an area of 4×10^5 m^2 within 3 hrs; Table II uses a total of 328 Pioneer robots to cover an area of 4×10^6 m^2 within 3 hrs. There are four groups in the first deployment and two groups in the second deployment. This is because the PPRK and Pioneer robots have different specifications, and also these two deployments cover two areas with different sizes. We can see from both tables that the later deployed groups have smaller numbers of robots and cover smaller areas. Corresponding to the two deployments, the equal-number-40 uses 320 PPRK robots and 448 Pioneer robots, respectively. Our method saves 14% PPRK robots and 36% Pioneer robots.

C. Robot Deployment in Areas With Obstacles

The robots' detouring around obstacles increases the traveling distance to cover an area with obstacles. Equation (8) shows the empirical rule we use in this paper to estimate the relationship between r_d and d_o. We use the constant $\alpha = 3$, as explained in the previous section. Figs. 21 and 22 show the fleet size at different obstacle densities. Fig. 21 is for PPRK robots covering an area of 3×10^5 m^2 with $\tau = 3$ hrs, and Fig. 22 is for Pioneer robots covering an area of 3×10^6 m^2 with $\tau = 3$ hrs. In both figures, the fleet size increases as the obstacle density increases, and our method uses the fewest robots. SPACA uses 200 PPRK robots at $d_o = 0$ and 280 PPRK robots at $d_o = 0.1$, while equal-number-40 uses 212 PPRK robots at $d_o = 0$ and 316 PPRK robots at $d_o = 0.1$. SPACA uses 6% and 12% fewer robots than equal-number-40, respectively. For Pioneer robots, SPACA uses 15% and 32% fewer robots than equal-number-40 at $d_o = 0$ and $d_o = 0.1$.

VIII. Conclusion

This paper presents a robot deployment strategy that can reduce the deployment overhead, and use a smaller number of robots to cover the same area. Our approach considers both energy and timing constraints, and determines a speed at which the robot can travel the longest distance. We use two obstacle models and derive an empirical rule for the extra detouring distance. Our deployment method is applied to covering areas with obstacles. Simulations are conducted on two robots called PPRK and Pioneer 3DX, and show that our method can reduce the total number of robots by up to 30%.

For future work, this study can be extended in the following three aspects. 1) We do not consider the sensing uncertainty in this paper. Sensing uncertainty can be considered by changing the sensing models. An area is considered as covered only when the confidence is higher than a threshold. 2) Our deployment algorithm adopts a divide-and-conquer strategy, and requires little communication. This can be extended to consider dynamic robot coordination where the robots communicate with each other when they are working. This could potentially increase the coverage efficiency. 3) We can consider some other existing coverage algorithms, and compare our scanline coverage algorithm with them.

References

[1] D. Auguelov, D. Koller, E. Parker, and S. Thrun, "Detecting and modeling doors with mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 3777–3784.

[2] R. Aylett, *Robots: Bringing Intelligent Machines to Life*. Hauppauge, NY: Barrons Educ. Ser., 2002.

[3] J. Baltes and J. Anderson, "Flexible binary space partitioning for robotic rescue," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2003, pp. 3144–3149.

[4] A. Barili, M. Ceresa, and C. Parisi, "Energy-saving motion control for an autonomous mobile robot," in *Proc. Int. Symp. Ind. Electron.*, 1995, pp. 674–676.

[5] M. A. Batalin and G. S. Sukhatme, "Efficient exploration without localization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2003, pp. 2714–2719.

[6] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. Int. Symp. Comput. Arch.*, 2000, pp. 83–94.

[7] H. J. Chang, C. G. Lee, Y.-H. Lu, and Y. C. Hu, "A computational efficient SLAM algorithm based on logarithmic-map parititioning," in *Proc. Int. Conf. Intell. Robots Syst.*, 2004, pp. 1041–1046.

[8] H. J. Chang, C. S. G. Lee, Y.-H. Lu, and Y. C. Hu, "Energy-time-efficient adaptive dispatching algorithms for ant-like robot systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 3294–3299.

[9] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. K. Saluja, "Sensor deployment strategy for detection of targets traversing a region," *Mobile Netw. Appl.*, vol. 8, no. 4, pp. 453–461, 2003.

[10] *"DC Motors, Speed Controls, Servo Systems, An Engineering Handbook,"* E.-C. Corp., 1973.

[11] S. Das, Y. C. Hu, C. S. G. Lee, and Y.-H. Lu, "Supporting many-to-one communication in mobile multi-robot ad hoc sensing networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 659–664.

[12] A. Davids, "Urban search and rescue robots: From tragedy to technology," *IEEE Intell. Syst.*, vol. 17, no. 2, pp. 81–83, Mar. 2002.

[13] F. Dellaert and A. W. Stroupe, "Linear 2D localization and mapping for single and multiple robot scenarios," in *Proc. ICAR*, 2002, pp. 688–694.

[14] P. Sanchez [Online]. Available: http://planetmath.org/encyclopedia/jensensinequality.html

[15] I. Duleba and J. Z. Sasiadek, "Nonholonomic motion planning based on Newton algorithm with energy optimization," *IEEE Trans. Control Syst. Technol.*, vol. 11, no. 3, pp. 355–363, May 2003.

[16] D. Erickson, "Non-learning artificial neural network approach to motion planning for the pioneer robot," in *Proc. Int. Conf. Intell. Robots Syst.*, 2003, pp. 112–117.

[17] J. Evans, "Helpmate: An autonomous mobile robot courier for hospitals," in *Proc. Int. Conf. Intell. Robots Syst.*, 1994, pp. 1695–1700.

[18] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multirobot coordination," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 758–768, Oct. 2002.

[19] E. Gonzalez, M. Alarcon, P. Aristizabal, and C. Parra, "BSA: A coverage algorithm," in *Proc. Int. Conf. Intell. Robots Syst.*, 2003, pp. 1679–1684.

[20] A. Howard, M. J. Mataric, and G. S. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Auton. Robots*, vol. 13, no. 2, pp. 113–126, 2002.

[21] T. Ishida, Y. Kuroki, and T. Takahashi, "Analysis of motions of a small biped entertainment robot," in *Proc. Int. Conf. Intell. Robots Syst.*, 2004, pp. 142–147.

[22] R. Katoh, O. Ichiyama, T. Yamamoto, and F. Ohkawa, "A real-time path planning of space manipulator saving consumed energy," in *Proc. Int. Conf. Ind. Electron., Control, Instrum.*, 1994, pp. 1064–1067.

[23] B. C. Kuo and J. Tal, Eds., *DC Motors and Control Systems*. New York: SRL Publishing, 1978.

[24] D. Kurabayashi, J. Ota, T. Arai, and E. Yoshida, "Cooperative sweeping by multiple mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1996, pp. 1744–1749.

[25] J. Liu, P. H. Chou, N. Bagherzadeh, and F. Kurdahi, "Power-aware scheduling under timing constraints for mission-critical embedded systems," in *Proc. Des. Autom. Conf.*, Jun. 2001, pp. 840–845.

[26] J. R. Lorch and A. J. Smith, "Apple Macintosh's energy consumption," *IEEE Micro*, vol. 18, no. 6, pp. 54–63, Nov. 1998.

[27] C. Luo and S. X. Yang, "A real-time cooperative sweeping strategy for multiple cleaning robots," in *Proc. Int. Symp. Intell. Control*, 2002, pp. 660–665.

[28] T. Makimoto and Y. Sakai, "Evolution of low power electronics and its future applications," in *Proc. Int. Symp. Low Power Electron. Des.*, 2003, pp. 2–5.

[29] L. Matthies, E. Gat, R. Harrison, B. Wilcox, R. Volpe, and T. Litwin, "Mars microrover navigation: Performance evaluation and enhancement," in *Proc. Int. Conf. Intell. Robots Syst.*, 1995, pp. 433–440.

[30] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *Proc. INFOCOM*, 2001, pp. 1380–1387.

[31] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. S. G. Lee, "Determining the fleet size of mobile robots with energy constraints," in *Proc. Int. Conf. Intell. Robots Syst.*, 2004, pp. 1420–1425.

[32] ——, "Energy-efficient motion planning for mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 4344–4349.

[33] ——, "Deployment strategy for mobile robots with energy and timing constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 2827–2832.

[34] S. Michaud, A. Schneider, R. Bertr, P. Lamon, R. Siegwart, M. V. Winnendael, and A. Schiele, "SOLERO: Solar powered exploration rover," in *Proc. 7th ESA Workshop Adv. Space Technol. Robot. Autom.*, 2002, CD-ROM.

[35] S. Poduri and G. S. Sukhatme, "Constrained coverage for mobile sensor networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 165–171.

[36] G. Reshko, M. T. Mason, and I. R. Nourbakhk, Rapid prototyping of small robots Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-RI-TR-02-11, 2002.

[37] P. E. Rybski, N. P. Papanikolopoulos, S. A. Stoeter, D. G. Krantz, K. B. Yesin, M. Gini, R. Voyles, D. F. Hougen, B. Nelson, and M. D. Erickson, "Enlisting rangers and scouts for reconnaissance and surveillance," *IEEE Robot. Autom. Mag.*, vol. 7, no. 4, pp. 14–24, Dec. 2000.

[38] K. Schreiner, "Landmine detection research pushes forward, despite challenges," *IEEE Intell. Syst.*, vol. 17, no. 2, pp. 4–7, Mar. 2002.

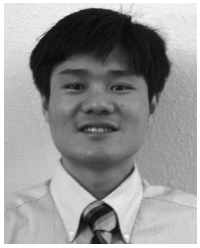
[39] R. Simmons, D. Apfelbaum, D. Fox, R. P. Goldman, K. Z. Haigh, D. J. Muslineg, M. Pelican, and S. Thrun, "Coordinated deployment of multiple, heterogeneous robots," in *Proc. Int. Conf. Intell. Robots Syst.*, 2000, pp. 2254–2260.

[40] T. Simunic, L. Benini, and G. D. Micheli, "Cycle-accurate simulation of energy consumption in embedded systems," in *Proc. Des. Autom. Conf.*, 1999, pp. 867–872.

[41] Z. Sun and J. Reif, "On energy-minimizing paths on terrains for a mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2003, pp. 3782–3788.

[42] S. Tadokoro, T. Takamori, K. Osuka, and S. Tsurutani, "Investigation report of the rescue problem at Hanshin-Awaji earthquake in Kobe," in *Proc. Int. Conf. Intell. Robots Syst.*, 2000, pp. 1880–1885.

[43] C. J. Taylor and D. J. Kriegman, "Vision-based motion planning and exploration algorithms for mobile robots," *IEEE Trans. Robot. Autom.*, vol. 14, no. 3, pp. 417–426, Jun. 1998.
[44] G. Wang, G. Cao, and T. L. Porta, "Movement-assisted sensor deployment," in *Proc. INFOCOM*, 2004, pp. 2469–2479.
[45] G. Xing, C. Lu, R. Pless, and Q. Huang, "On greedy geographic routing algorithms in sensing-covered networks," in *Proc. MobiHoc*, 2004, pp. 31–42.
[46] H. Yamaguchi, "Adaptive formation control for distributed autonomous mobile robot groups," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1997, pp. 2300–2305.
[47] F. Yamasaki, K. Hosoda, and M. Asada, "An energy consumption-based control for humanoid walking," in *Proc. Int. Conf. Intell. Robots Syst.*, 2002, pp. 2473–2477.
[48] A. Zelinsky, "A mobile robot exploration algorithm," *IEEE Trans. Robot. Autom.*, vol. 8, no. 6, pp. 707–717, Dec. 1992.
[49] Y. Zhang, M. Schervish, E. U. Acar, and H. Choset, "Probabilistic methods for robotic landmine search," in *Proc. Int. Conf. Intell. Robots Syst.*, 2001, pp. 1525–1532.
[50] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," in *Proc. INFOCOM*, 2003, pp. 1293–1303.



Yongguo Mei (S'04) is currently working toward the Ph.D. degree in the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN. He received the M.S. degree from ShangHai Jiao Tong University, Shanghai, China, in 2001.

His research focuses on energy-efficient mobile autonomous systems, including mobile robots, embedded systems, and wireless systems.

Yung-Hsiang Lu (S'90–M'03) received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 2002.

He currently is an Assistant Professor with the School of Electrical and Computer Engineering, and by courtesy, the Department of Computer Science, at Purdue University, West Lafayette, IN. His research focuses on power management and energy conservation for computer systems, mobile robots, sensor networks, and image processing.

In 2004, Dr. Lu received an NSF Career Award for studying energy conservation by operating systems.

Y. Charlie Hu (S'91–M'98) received the M.S. and M.Phil. degrees from Yale University, New Haven, CT, in 1992, and the Ph.D. degree in computer science from Harvard University, Cambridge, MA, in 1997.

Currently, he is an Assistant Professor of Electrical and Computer Engineering and Computer Science, Purdue University, West Lafayette, IN. From 1997 to 2001, he was a Research Scientist with Rice University, Houston, TX. His research interests include operating systems, distributed systems, networking, and parallel computing. He has published more than 65 papers in these areas.

Dr. Hu received the NSF CAREER Award in 2003. He served as a TPC Vice Chair for the 2004 International Conference on Parallel Processing (ICPP-04), and a Co-founder and TPC Co-Chair for the First International Workshop on Mobile Peer-to-Peer Computing (MP2P'04). Dr. Hu is a member of USENIX and ACM.

C. S. George Lee (S'71–M'78–SM'86–F'93) received the B.S. and M.S. degrees in electrical engineering from Washington State University, Pullman, in 1973 and 1974, respectively, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 1978.

He is a Professor of Electrical and Computer Engineering at Purdue University, West Lafayette, Indiana. Currently, he is serving as Program Director for the Robotics Program at the National Science Foundation (NSF). His current research focuses on humanoid robots, robotic SLAM, and neuro-fuzzy systems. He has authored or co-authored over 150 publications in these areas, in addition to two graduate textbooks: *Robotics: Control, Sensing, Vision, and Intelligence* (New York: McGraw-Hill, 1986) and *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems* (Englewood Cliffs, NJ: Prentice-Hall, 1996), and 20 book chapters.

Dr. Lee was an IEEE Computer Society Distinguished Visitor in 1983–1986, and the Organizer and Chairman of the 1988 NATO Advanced Research Workshop on Sensor-Based Robots: Algorithms and Architectures. He had also served as Secretary and Vice-President for Technical Affairs of the IEEE Robotics and Automation Society in 1988–1990 and 1990–1995, respectively. He was General Chair of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems in Las Vegas, NV, and was Program Chair of the 1996 IEEE International Conference on Robotics and Automation in Minneapolis, MN, and the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems in Victoria, BC, Canada. Currently, he serves as General Co-Chair of the 2006 IEEE International Conference on Robotics and Automation, held in Orlando, FL, in May 2006. He is a recipient of The IEEE Third Millennium Medal Award and the Distinguished Service Award from the IEEE Robotics and Automation Society.