Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

Deployment Strategy for Mobile Robots with Energy and Timing Constraints

Yongguo Mei, Yung-Hsiang Lu, Y. Charlie Hu, and C. S. George Lee
School of Electrical and Computer Engineering, Purdue University
{ymei, yunglu, ychu, csglee}@purdue.edu

***Abstract*— Mobile robots usually carry limited energy and have to accomplish their tasks before deadlines. Examples of these tasks include search and rescue, landmine detection, and carpet cleaning. Many researchers have been studying control, sensing, and coordination for these tasks. However, one major problem has not been fully addressed: the initial deployment of mobile robots. The deployment problem considers the number of robots needed and their initial locations. In this paper, we present a solution for the deployment problem when robots have limited energy and time to collectively accomplish coverage tasks. Simulation results show that our method uses 26% fewer robots comparing with two heuristics for covering the same size of area.**

I. Introduction

Mobile robots can be used in many applications, and usually carry limited energy, such as batteries. Thus, energy constraints limit the operational time of mobile robots. Meanwhile, many tasks have timing constraints. For example, search and rescue usually has to find survivors within 24 hours; otherwise, the chance of survival diminishes quickly. Another example is to detect and destroy landmines before troops arrive. Energy and time can be conflicting constraints. For example, a vehicle can travel at a high speed and reach the destination earlier (meeting the timing constraint). However, fuel efficiency (miles per gallon) can drop dramatically at a high speed and the vehicle may run out of fuel (failing the energy constraint). It is crucial to consider both constraints together. 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 deploying mobile robots to address two issues: (a) the number of robots needed (i.e. "fleet size") to search an area and (b) the initial locations of these robots.

We can use survivor detection after an earthquake to explain the deployment problem. Small robots can move under the 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 to help rescuers find survivors. The deployment problem is affected by each robot's energy capacity, the deadline, and the moving speed. A desirable deployment strategy should meet the following goals: (a) 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. (b) It can cover the area within the energy and the timing constraints.

Our deployment approach considers the time spent in unloading the robots from the carrier. This unloading time accounts for the time to remove the robots from the carrier, put them on the ground, and instruct the robots to start moving. Because all robots have the same deadline to finish the task, a robot that is unloaded latter has shorter time before the deadline. After the robots are unloaded, they disperse from the unloading location and reach the individual regions where the robots are responsible for searching. If more robots are unloaded at the same location, some robots will waste much time and energy to move from the unloading location to the starting locations of their regions. We consider three types of overhead during the deployment: unloading time, dispersing time, and partially overlapped regions (explained later in the paper). Our method is called Space Partition Area Coverage Algorithm (SPACA). Simulation results show that our method uses 26% fewer of the robots to cover the same areas when it is compared with two kinds of heuristics.

II. Previous Work

Energy conservation is an important issue for mobile robots. Meanwhile, many tasks have timing constraints. Aylett [1] points out that energy constraints are the most important challenge for mobile robots. Barili et al.

[1]This work was supported in part by the National Science Foundation IIS-0329061 and Career CNS-0347466.

[3] control the velocities to save energy for a mobile robot. Mei et al. [9] present an energy model for mobile robots and discuss the energy properties of three motion patterns: scanline, spiral and square spiral. Multiple robots can cooperate to accomplish tasks, such as search and rescue, carpet cleaning, and landmine detection. Baltes et al. [2] show a flexible space partition method for robot rescue. Das et al. [7] compare communication schemes among mobile robots. Zhang et al. [13] use a probabilistic method for searching landmines.

To use multiple robots efficiently, they have to be deployed at the proper locations. Simmons et al. [12] study the multi-robot coordination using robot deployment as an example. Their paper focuses on the control and coordination. Rybski et al. [11] use large "ranger" robots to transport and deploy small "scout" robots. The rangers can travel up to 20 kilometers, greatly extending the search range of scouts. Chang et al. [5] study the energy and time properties of different dispatching algorithms for ant-like robot systems. The ant-like robots start from a nest to explore unmapped terrains. Cloqueur et al. [6] discuss the sensor deployment strategy. Considering both the deployment cost and the sensor cost, they provide a sequential deployment process; however, they do not consider the timing constraints. An efficient deployment can decrease the deployment overhead, thus reducing the number of robots needed. Mei et al. [8] present a probabilistic model to determine the number of robots for serving random pickup-delivery requests with timing and energy constraints.

III. Problem Description

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

(a) All robots are the same; they have the same amount of initial energy $\mathcal{E}$. Each robot's power consumption is affected only by its speed. All the robots have the same deadline; i.e., they have to finish their jobs before the same time.

(b) 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 the robot's traveling distance.

(c) The area to be covered is a two-dimensional region without obstacles. The robots travel along scanlines to cover the area; the time and energy for changing directions are not considered. A detailed analysis of the energy consumption of scanlines is presented in our previous study [9].

(d) 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.

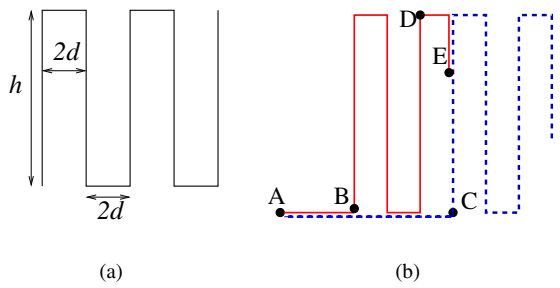
We consider only one carrier in this paper but our method can be extend to multiple carriers. At time $t = 0$, the carrier starts moving to the first unloading site. The area has to be covered before the deadline $t = \tau$. The carrier unloads robots at k different locations. At the i^{th} location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

IV. Deployment Strategy

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

A. Mobile Robots' Energy Models

We consider the robots' motion power only— the sum of the power consumed by the motors. A DC motor's power consumption is primarily due to the output mechanical power and the armature loss [4]. We use $p(v)$ to represent the power consumption at speed v, within the maximum speed v_m. The energy efficiency can be defined as the distance traveled with one unit of energy, and the energy efficiency at speed v is $\frac{v}{p(v)}$. Higher efficiency means less energy for the same distance. We use v_o ($0 < v_o < v_m$) to represent the speed of the highest efficiency: $\frac{v_o}{p(v_o)} \geq \frac{v}{p(v)}$, $\forall v, 0 < v < v_m$. In this paper, we assume all the robots move at the optimal speed v_o. A detailed discussion of the effects of speed and energy efficiency is available in our previous study [9].



(a) (b)

Fig. 1. (a) 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.

B. Overhead in Deployment

There are several ways to cover an area, as discussed in our previous study [9]. This paper considers scanlines only. Figure 1 (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.

(a) The first type of overhead is the time spent for unloading the robots.
(b) The second type of overhead comes from the time and the energy spent by each robot to reach its starting location after being unloaded.
(c) 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 fragmentation of hard disks.

They are called unloading, dispersing, and fragmentation overhead in the rest of this paper. These types of overhead are related. Figure 1 (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 has zero 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 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.

C. 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 two-dimensional Cartesian coordinates centered from the unloading location. Figure 2 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 5^{th} and 6^{th} robots. Point C is the starting location of next two robots— the 7^{th} and the 8^{th}. Because the 5^{th} 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.

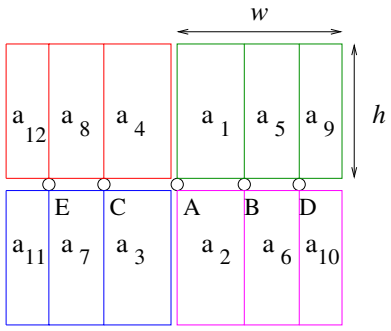


Fig. 2. The area is covered by a group of 12 robots. The areas subscribed from 1 to 12 are the areas covered by these robots. The areas are symmetric to the unloading location A.

Let w be the width of the area covered by one quarter of the robots in the same group, as shown in Figure 2. The total dispersing distance of the three robots covering a_1, a_5 and a_9 is $0 + \overline{AB} + \overline{AD} \approx 0 + \frac{w}{3} + \frac{2w}{3} = w$. 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 $\frac{w}{\psi} + \frac{2w}{\psi} + ... + \frac{(\psi-1)w}{\psi} = \frac{(\psi-1)w}{2}$. The average dispersing distance for each robot is $\frac{(\psi-1)w}{2}\frac{1}{\psi} \approx \frac{w}{2}$. Meanwhile, the average fragmentation overhead for each robot is $\frac{h}{2}$. 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 $\frac{\mathcal{E}}{p}$ so the robot can operate at most $\min(\frac{\mathcal{E}}{p}, \tau)$ and travel at most $v \times \min(\frac{\mathcal{E}}{p}, \tau)$. 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 $\frac{wh}{2d}$. This area is reduced due to the overhead so $\psi l \approx \frac{wh}{2d} + \psi(\frac{w}{2} + \frac{h}{2})$. 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 $\psi l \approx \frac{h^2}{2d} + \psi h$. 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.

D. Number of Groups and Group Sizes

To minimize the number of robots used is equivalent to maximize 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. Based on these observations, 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 i^{th} and the j^{th} 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 i^{th} group and more robots in the j^{th} 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 of a deployment. The next section presents our algorithm generating deployment solutions that satisfy the above three rules.

E. Deployment Algorithm

```
ROBOT-DEPLOYMENT(E, τ, A)
 1  E_t ← E, τ_t ← τ, A_t ← A
    /* E: energy, τ: deadline, A: total area to cover */
 2  n_1 ← 1
 3  n_p ← n_1
 4  τ ← τ − u(n_p)
 5  A ← A − GET-TOTAL-AREA(E, τ, n_p)
 6  min-area ← GET-MINIMUM-AREA(E, τ, n_p)
 7  flag ← 0 /* no solution yet */
 8  while flag = 0
 9        do n ← n_p /* n is the fleet size */
10           diff ← ∞ /* initialization */
11           while A > 0 and τ > 0
12                 do for i ← 1 to n_p /* select next group size */
13                          do τ ← τ − u(i)
14                             temp ← GET-MINIMUM-AREA(E, τ, i)
15                             if i = 1 and temp < min-area
16                               then τ ← −1, A ← 1
17                                    break
18                             /* find the closest minimum area */
19                             if | min-area − temp | < diff
20                               then diff = | min-area − temp |
21                                    x ← i
22                             τ ← τ + u(i)
23                    τ ← τ − u(x)
24                    A ← A − GET-TOTAL-AREA(E, τ, x)
25                    n ← n + x
26                    n_p ← x
27                    min-area ← GET-MINIMUM-AREA(E, τ, x)
28           if τ > 0 /* before the deadline */
29             then flag ← 1
30             else n_1 ← n_1 + 1 /* change first group size */
31                  E ← E_t, τ ← τ_t, A ← A_t
32                  n_p ← n_1
33                  τ ← τ − u(n_p)
34                  A ← A − GET-TOTAL-AREA(E, τ, n_p)
35                  min-area ← GET-MINIMUM-AREA(E, τ, n_p)
    /* adjust the size of last group */
36  n ← n − x
37  A ← A + GET-TOTAL-AREA(E, τ, x)
38  τ ← τ + u(x)
39  for j ← 1 to x
40        do τ ← τ − t_u(j)
41           if A ≤ GET-TOTAL-AREA(E, τ, j)
42             then n ← n + j
43                  break
44             else τ ← τ + u(j)
45  return n
```

Our algorithm 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 that the algorithm may leave the inner **while** loop. (a) The first case happens in the first **if** statement inside the inner **while** loop. When the next group size is 1 and the minimum area is less than the previous minimum area, the first rule is impossible to be fulfilled. Therefore, we assign a negative value to τ to leave the inner **while** loop. (b) In the second case, the time left τ is non-positive, while the left area $\mathcal{A}$ is still positive. This means the area has not been fully covered but no time is left. (c) The third case happens when the area is covered and there is still time. In cases (a) and (b), 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 area left before unloading the last group, we may use more robots than necessary to cover the whole area in the last group. The last steps adjust the size based on the area left for the last group. This algorithm will report the situation when it is impossible to cover the area under the constraints.

V. A Case Study

A. Simulation Setup

A commercial robot called PPRK is used for our case study. The robot is developed at Carnegie Mellon University [10]. It has three polyurethane omni-directional wheels driven by three MS492MH DC servo motors. The energy capacity is $20736J$ for four AA batteries (1200 mAh and 1.2V). A data acquisition card is used to measure the voltage and current to calculate the robot's power at different speeds [9]. The power model used in this paper is $p(v) = 48.31v^2 - 3.37v + 0.69$. The maximum speed is $0.16m/s$, and the optimal speed is $0.12m/s$ with power consumption 0.98W. The unloading time is $u(n) = 600 + 2.5n$ seconds for n robots. The sensing distance used is $0.8m$.

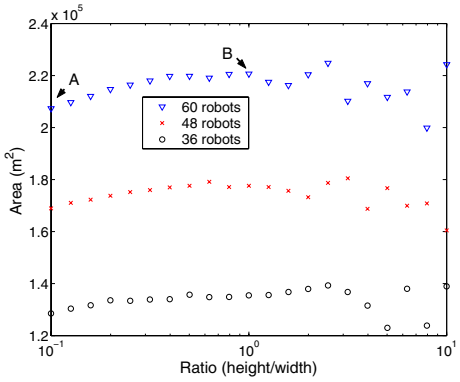


Fig. 3. Area covered by different number of robots with different ratios of height and width

B. Area Covered by One Group

Figure 3 shows the size of the area covered by different numbers of robots with different height-width ratios. The robots have 6 hours before the deadline. 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 6%. 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 very 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 $10w$. However, the same ratio $h = 10w$ can cover a smaller area with 48 robots. Hence, we choose $h \approx w$ in our algorithm because this provides stably large covered areas.

C. Simulation Results

We compare our method with two heuristics. The first is equal-number deployment. This method unload equal number of robots each time until the assigned area can be covered. The number of robots in each group is decided before the deployment. The second unloads all robots at one location. We call this one-unloading method. This method saves the unloading time but increases the dispersing overhead.

Figures 4 and 5 have deadlines (τ) four and eight hours, respectively. For equal-number deployment, we choose two different numbers, 4 and 10. In both 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 (Figure 4). In Figure 4, equal-number (4) needs the most robots and it can cover at

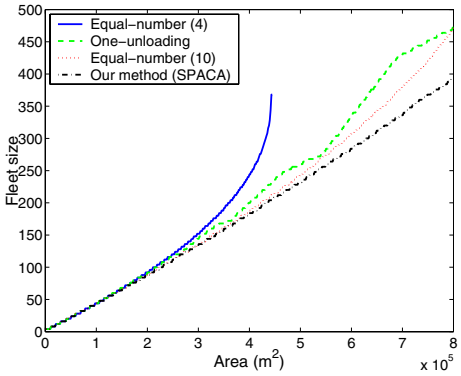


Fig. 4. Fleet size verses area, τ =4hours, energy = 20736J

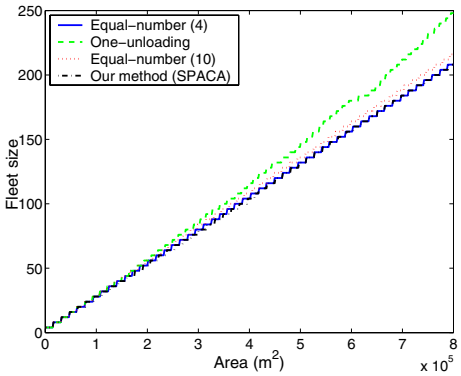


Fig. 5. Fleet size verses area, τ =8hours, energy = 20736J

most $4.45\times10^5m^2$. The last group is deployed very close to the deadline and no more groups can be deployed before the deadline. Figure 5 allows a longer deadline so unloading time is less important. Deployments with a small group size can save dispersing and fragmentation overhead, thus requiring fewer robots than the deployments with a large group size. Equal-number (4) needs almost the same number of robots as our method, while one-unloading needs the most robots. Table I shows the details of the deployment generated by our method to cover an area of $6.8 \times 10^5m^2$ within 4 hours. This deployment uses 328 robots. With the same conditions, the one-unloading method has to use 416 robots, and the average area per robot is $1634m^2$. Our method has a higher average area than that of the one unloading method, and saves more than 26% of the robots.

VI. Conclusion

This paper presents a method to deploy mobile robots for covering an area with energy and timing constraints. Our approach determines the number of robots in each group and the number of groups. The principles in our approach are making the width equal to the height in each group and making the minimum area covered by each group the same. We use a case study to demonstrate the effectiveness of our method. For future work, we plan to extend this research in two aspects: (1) considering obstacles, and (2) including speed variations.

Group number	1	2	3	4	5
Group size	120	84	64	52	8
Minimum area	1783	1808	1836	1876	1936
Average area	2214	2118	2008	1892	1936
Total area	265680	177912	128512	98384	15488

TABLE I

Deployment for covering $6.8 \times 10^5m^2$ within four hours.

References

[1] R. Aylett. *Robots: Bringing Intelligent Machines To Life*. Barrons, 2002.

[2] J. Baltes and J. Anderson. Flexible Binary Space Partitioning for Robotic Rescue. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3144–3149, 2003.

[3] A. Barili, M. Ceresa, and C. Parisi. Energy-Saving Motion Control for An Autonomous Mobile Robot. In *International Symposium on Industrial Electronics*, pages 674–676, 1995.

[4] P. Bertoldi, A. de Almeida, and H. Falkner. *Energy Efficiency Improvements in Electric Motors and Drives*. Springer, 2000.

[5] 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 *International Conference on Robotics and Automation*, pages 3294–3299, 2004.

[6] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. K. Saluja. Sensor Deployment Strategy for Target Detection. In *International Workshop on Wireless Sensor Networks and Applications*, pages 42–48, 2002.

[7] 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 *International Conference on Robotics and Automation*, pages 659–664, 2004.

[8] Y. Mei, Y.-H. Lu, C. S. G. Lee, and Y. C. Hu. Determining the Fleet Size of Mobile Robots with Energy Cons traints. In *IEEE / RSJ International Conference on Intelligent Robots and Systems*, pages 1420–1425, 2004.

[9] Y. Mei, Y.-H. Lu, C. S. G. Lee, and Y. C. Hu. Energy-Efficient Motion Planning for Mobile Robots. In *International Conference on Robotics and Automation*, pages 4344–4349, 2004.

[10] G. Reshko, M. T. Mason, and I. R. Nourbakhk. Rapid Prototyping of Small Robots. Technical report, CMU-RI-TR-02-11, Carnegie Mellon University, 2002.

[11] 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 Robotics and Automation Magazine*, 7(4):14–24, December 2000.

[12] 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 *International Conf. on Intelligent Robots and Systems*, pages 2254–2260, 2000.

[13] Y. Zhang, M. Schervish, E. U. Acar, and H. Choset. Probabilistic Methods for Robotic Landmine Search. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1525–1532, 2001.