Dynamic Decentralized Area Partitioning for Cooperating Cleaning Robots

Markus Jäger
Corporate Technology
Siemens AG
81730 Munich, Germany
markus.jaeger@mchp.siemens.de

Bernhard Nebel
Institut für Informatik
Albert-Ludwigs-Universität
79100 Freiburg, Germany
nebel@informatik.uni-freiburg.de

Abstract

If multiple cleaning robots are used to cooperatively clean a larger room, e.g. an airport, the room must be partitioned among the robots. This paper describes a dynamic and decentralized method to partition a certain area among multiple robots.

The area is divided into polygons, which are allocated by the robots. After a robot has allocated a certain polygon, it is responsible for cleaning the polygon.

The method described in this paper does not need any global synchronization and does not require a global communication network.

1 Introduction

If multiple cleaning robots [1] are used to cooperatively clean a larger room, e.g. an airport, the room must be partitioned among the robots. This can be done either statically [2] or dynamically.

A static approach assigns each robot a certain subarea at the beginning. Each robot is then only responsible for its assigned subarea. The main disadvantage of a static assignment is that the whole system can not be adapted dynamically to a new situation. If, for example, a robot breaks down, the other robots can not take over its work, or if a robot is slower than assumed, the other robots are not able to help him.

A dynamic partition, however, assigns the subareas during runtime. It is therefore possible to react to unpredictable events. If, for example, a robot breaks down, the other robots can dynamically take over his work. The main disadvantage is that the robots have to communicate regularly, in order to cooperate.

Our method, described in the following, uses a completely dynamic approach. The main idea is to divide the room into polygons, as shown in Figures 1 and 2. The robots then allocate and clean these polygons. To allocate a polygon means that a robot intends to clean the polygon and announces this.

In contrast to other dynamic approaches (see chapter 2) it is not assumed that the robots are always able to communicate with each other. There is therefore no need for a global communication network. For the algorithms to work properly, it is sufficient if the robots can communicate every now and then, in oder to exchange information.

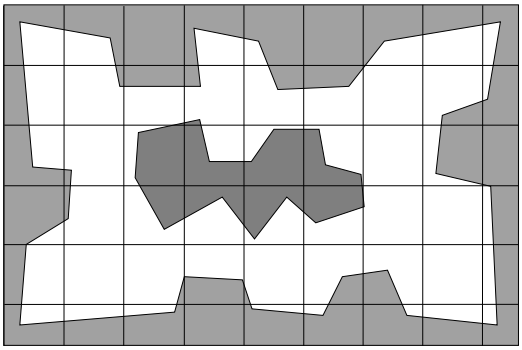

Figure 1: Division of an area into polygons.

Some related work, which also concentrates on area partitioning, is summarized in the next section. Section three describes the method itself and the last two sections provide some simulation results and give a summary.

2 Related Work

Besides a few exceptions (see below), all the methods for area partitioning use static approaches. The main difference is how the subareas are determined.

Hert *et. al.* [2, 3] deal with the question of how to split an area in n parts (for n robots), which are equal in size, where each part has to be connected and has to contain a certain point. These points correspond to the robots start positions.

The results of Hert *et. al.* are mathematically sound, but often provide solutions which are unuseable for real applications. Bern *et. al.* [4] therefore try to find areas of equal size which have no acute angles, so that a robot can process them more easily.

Christou *et. al.* [5, 6, 7], in contrast, try to find subareas which have a minimal diameter, since this results in subareas with a very small perimeter. They state that a small

perimeter reduces the risk of collisions, since the robots are more seldom at the edge of their area.

There are two different dynamic approaches. One is from Hert *et. al.* [8]. It differs from their static approach, described above, only in the number of parts in which the area is split. The area is not split in n parts, but in $n+1$. The last part is distributed dynamically after the robots have finished their initial parts. How the dynamic distribution is done, however, is not described.

The other dynamic approaches are by Schneider-Fontán and Matric [9], and by Min and Yin [10]. Their idea is also to split the area in n parts and to assign each robot one part at the beginning. The robots can, however, negotiate and dynamically redistribute subparts while they are processing their parts. In contrast to our dynamic approach, their algorithms require a global communication network.

3 Dynamic Area Partitioning

As stated above, the method described in this paper uses a dynamic approach. After the whole area is separated into polygons (Section 3.1), single polygons are chosen (Section 3.3), with the help of a connectivity graph (Section 3.2), and allocated (Section 3.4). After a robot has allocated a polygon, it starts to clean the polygon. (In fact, robots can allocate more than one polygon and process them simultaneously, if it is advantageous for the cleaning strategy.)

To actually clean a polygon, various approaches can be taken. This is, however, not part of this work. The work of Hofner and Schmidt [11], and Acar *et. al.* [12] could be a starting point for the interested reader.

If a robot has allocated or cleaned a polygon, it has to inform the other robots (Section 3.5), since the other robots would otherwise also allocate and clean the polygon. To inform other robots is, however, not always possible, since the robots are not always able to communicate. In the following, it is assumed that two robots are able to communicate whenever their distance is lower than r_{com} (r_{com} therefore represents the communication radius of the robots). Figure 3 shows three robots with circles around them. The circles have a radius of $\frac{r_{com}}{2}$, which means that two robots are able to communicate, if their respective circles intersect.

Each robot works until it knows that all polygons are allocated, and until it has cleaned all polygons which itself has allocated.

Since it it possible that the robots meet each other, special care has to be taken in order to avoid collisions among the robots. For that purpose a special decentralized method is used [13].

Some possible extensions are described in Section 3.6.

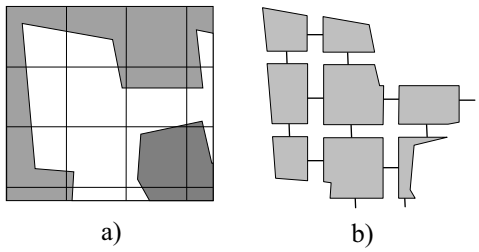

Figure 2: Figure a) shows the upper left region of the area and Figure b) shows the respective polygons. Figure b) furthermore shows the edges of the connectivity graph for the upper left region.

3.1 Separation into Polygons

To separate the area into polygons, it is overlaid with a grid (Figure 1). This leads to a number of polygons whose edges are based on the edges of the area and the lines of the grid. Figure 2 a) shows the upper left region of the area and Figure 2 b) shows the respective polygons.

The polygons are used to distribute the area among the robots. The robots successively allocate and clean the polygons.

3.2 Construction of a Connectivity Graph

To construct a connectivity graph, for each polygon a node is constructed and each two nodes, whose respective polygons have a common border, are connected by an edge. The connectivity graph gives information about how the single polygons are connected with each other. Figure 2 b) shows the connectivity graph of the upper left region of the area.

With each graph node some information about the associated polygon is stored, e.g. whether a polygon is already allocated or cleaned.

A connectivity graph is a well known concept [14]. It is, however, mainly used for navigation tasks.

3.3 Selection of Polygons for Allocation

The polygons which a robot allocates are chosen based on two principles. On the one hand, the robots make sure that they have always a few polygons in stock (Section 3.3.1). In stock means that the robot has already allocated a polygon, but not yet cleaned completely, i.e. it is currently processing the polygon. On the other hand, additional polygons are allocated, if it seems advantageous (Section 3.3.2).

A restriction is that a polygon can only be allocated if the robot is in range, i.e. the robot is less than $\frac{r_{com}}{2}$ away from the center of the polygon. This restriction is introduced to ensure that two robots, which want to allocate a polygon at the same time, are able to communicate with each other. (The case where two robots allocate the same

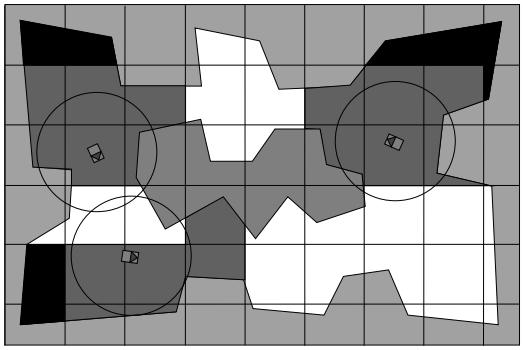

Figure 3: This figure shows three robots, where each robot has already cleaned some polygons (black regions) and is currently processing some polygons (dark gray regions). The circles around the robots have a radius of $\frac{r_{com}}{2}$. This means that two robots are able to communicate, if their respective circles intersect.

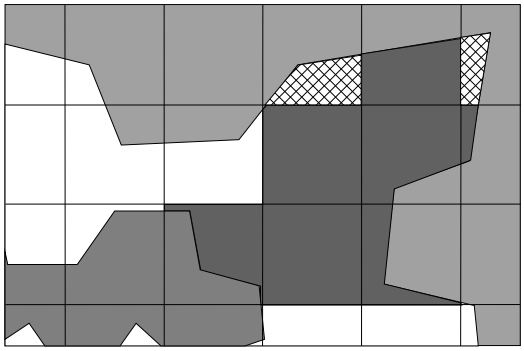

Figure 4: This figure shows two isolated (hatched) polygons.

polygon at different points in time is described in Section 3.5.)

If a robot has no polygons in stock any more and if it is not able to allocate a new one, because none is in range, it moves towards a new one until it is in range.

3.3.1 Stock Allocation

Since the robots make sure that they have always a few polygons in stock, they start to allocate new ones, whenever the number of polygons in stock falls below a certain threshold. Figure 3 shows three robots, where each robot has already cleaned some polygons (black regions) and has currently some polygons in stock (dark gray regions).

One restriction, however, is that new polygons are only allocated, if the currently stocked polygons are fully connected. This restriction is necessary to avoid jagged areas, which are processed at the same time. The connectivity of polygons is determined based on the connectivity graph (Section 3.2).

The selection of the polygons, which are allocated, is based on the following criteria:

- A polygon is only allocated, if no other robot (or the robot itself) is known to have already allocated the polygon.
- A new polygon must be connected to the currently processed polygons (the currently stocked polygons).
- The new polygon, in combination with the other stocked polygons, should result in an area with a very low diameter. This reduces the risk of collisions with other robots [5, 6, 7] and brings some advantages for the cleaning strategies, since they often favor a compact area.

3.3.2 Special Allocation

The idea of the special allocation is to avoid isolated polygons. A polygon is called isolated if it has no unallocated neighbor polygons any more. Figure 4 shows two isolated polygons (hatched polygons).

If such polygons would not be treated, it could happen that all polygons in the neighborhood of the isolated polygon are already allocated and cleaned, but not the isolated one. This in turn means that a robot might have to make a big detour, in order to finally allocate and clean the polygon.

Therefore, if a robot detects that an isolated polygon is next to its currently processed polygons, it allocates it.

3.4 Allocation of a Polygon

If a robot has chosen a polygon for allocation, it has to cooperate with all the other robots, which also want to allocate the polygon. It is assured that all the affected robots are able to communicate, since they all have to be in range of the polygon (see Section 3.3).

Before the robot starts to allocate the polygon, it computes the importance value (IV) of the polygon. The IV of the polygon indicates how important the allocation is (for the robot). A higher IV means that the allocation is more important. The IV of a polygon depends mainly on whether the new polygon, in combination with the other stocked polygons, results in an area with a low diameter (see also Section 3.3.1). The lower the diameter of the resulting area, the higher the IV. The IV is calculated by

$$IV(poly) = 1 - \frac{\sum_{p \in SP} \frac{min(dist(p,poly), MAXDIST)}{MAXDIST}}{|SP|}, \quad (1)$$

where SP is the set of stacked polygons, $dist$ computes the distance of the polygon centers, and $MAXDIST$ ist the maximum distance of two polygons (introduced to scale the IV to $[0 \ldots 1]$).

The allocation strategy is a combination of an election algorithm [15] and the *Contract Net*-protocol [16]. It comprises the following steps:

1. A robot, which wants to allocate a polygon, send an AllocationRequest message to all other robots with which it is able to communicate. The message contains the ID of the polygon and its IV.
2. If a robot receives an AllocationRequest message,
   - it sends an AllocationRefuse message, if itself has already sent an AllocationRequest message for the same polygon and if its IV is higher than the IV of the received AllocationRequest message (if the IVs are equal, the IDs of the robots are used to make a decision),
   - it sends an AllocationAccept message, if itself has already sent an AllocationRequest message for the same polygon and if its IV is lower than the IV of the received AllocationRequest message and
   - it sends an AllocationAccept message, if itself has not sent an AllocationRequest message for the same polygon.
3. If a robot receives an AllocationRefuse message, it knows that its allocation can not succeed and terminates it.
4. The allocation of a polygon is successful, if a robot receives an AllocationAccept message for each AllocationRequest message, which it has sent.
5. After a robot has allocated a new polygon, it informs all the other robots (see also Section 3.5).

During an allocation, special care has to be taken if new robots come into communication range or if robots leave the communication range. This means that additional AllocationRequest messages (step 1.) have to be sent to new robots and that it must not be waited for AllocationAccept messages (step 4.) of robots which already left the communication range.

3.5 Information Propagation

As stated above, the robots are not able to communicate with each other all the time. Therefore, the robots gradually develop their own local views, concerning allocated and cleaned polygons. If, for example, a robot allocates a new polygon and an other robot is not in communication range at that time, the other robot does not know anything about the allocation. It therefore has an other local view.

To synchronize their local views, the robots exchange information about allocated and cleaned polygons, whenever they are able to communicate. They thereby develop a common view. For example, if two robots, which have allocated the same polygon at different points in time (multiple allocations of the same polygon can happen if the allocating robot does not know that the polygon is already allocated by an other robot), meet, they exchange information about the polygon and therefore develop a common view concerning that polygon. If, for example, one robot has already cleaned the polygon, it tells this the other robot, which, in that case, stops to process the polygon and also marks it as cleaned.

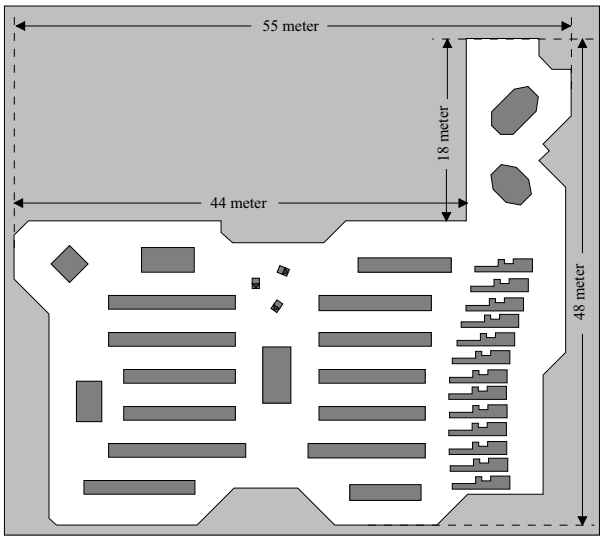

Figure 5: Supermarket in Bussum, Netherlands ($\approx 1812m^2$), shown with three robots.

The smaller the communication range r_{com} is, the less often are the robots able to communicate with each other and the more differ the local views of the robots. If the local views differ too much, it might happen that some polygons are allocated and cleaned multiple times. In the worst case each robot would allocate and clean all polygons. This is obviously not preferable. In Section 4 this is investigated in more detail.

3.6 Possible Extensions

Some possible extensions for the described method could be

- to introduce meeting points, at which the robots meet regularly, to synchronize their local views,
- the use of a specialized robot, which is not cleaning, but moves between the other robots all the time, in order to distribute information,
- to allow a robot to release certain already allocate polygons, if an other robot already finished its work and therefore could help the robot and process the polygons,
- to dynamically adapt the number of polygons which are allocated in advance (see Section 3.3.1), i.e. if there are only few unallocated polygons left, the number of polygons allocate in advance is also low (would reduce the deviation in processing times of the single robots), and

r_{com}	n	double polygons		double polygons (average)	
		allocated	processed	allocated	processed
13 meter	2 robots	0.9%	0.0%	0.9%	0.23%
		0.0%	0.0%		
		1.8%	0.7%		
	3 robots	0.0%	0.0%	0.0%	0.0%
		0.0%	0.0%		
		0.0%	0.0%		
	4 robots	0.0%	0.0%	0.0%	0.0%
		0.0%	0.0%		
		0.0%	0.0%		
10 meter	2 robots	0.0%	0.0%	1.3%	0.77%
		1.5%	0.9%		
		2.4%	1.4%		
	3 robots	0.0%	0.0%	0.43%	0.2%
		1.3%	0.6%		
		0.0%	0.0%		
	4 robots	0.0%	0.0%	0.0%	0.0%
		0.0%	0.0%		
		0.0%	0.0%		
7 meter	2 robots	3.8%	1.8%	3.63%	2.06%
		5.5%	2.8%		
		1.6%	1.6%		
	3 robots	1.7%	0.0%	2.67%	1.3%
		1.8%	1.2%		
		4.5%	2.7%		
	4 robots	2.8%	1.5%	2.3%	0.87%
		1.8%	0.7%		
		2.3%	0.4%		
4 meter	2 robots	16.3%	12.7%	17.53%	12.07%
		23.2%	21.1%		
		13.1%	2.4%		
	3 robots	10.4%	3.6%	14.43%	9.2%
		17.3%	13.1%		
		15.6%	10.9%		
	4 robots	14.8%	12.4%	13.8%	10.77%
		16.9%	13.6%		
		9.7%	6.3%		

Table 1: Simulation results for different communication ranges (r_{com}) and different numbers of robots (n). The simulation is based on a CAD map of a large supermarket in Bussum, Netherlands.

- to explicitly consider breakdowns of robots. With the current strategy, a breakdown of a robot is considered only implicitly, i. e. the broken down robot does not allocate polygons anymore, which means that the polygons, which the robot would have allocated and cleaned in the future, are automatically allocated and cleaned by other robots. Polygons which the robot has already allocated are, however, not processed by the other robots.

4 Simulation Results

The limited communication abilities of the robots lead to different local views of the robots, which might result in unnecessary work (see Chapter 3.5). This section describes a number of simulation runs, which were performed to investigate the influence of the local views on the performance of the whole system. The simulations are based on a CAD map of a large supermarket in Bussum, Netherlands. The supermarket is shown with three robots in Figure 5.

For different communication ranges and different numbers of robots, there have been made three simulation runs respectively. The size of the grid, which is used to separate the area into polygons (see Section 3.1), was set to 3.5 meters. Table 1 shows the simulation results, where the first two columns indicate the communication range (r_{com}) and the number of robots (n). Column 3 and 4 give the percentage of double allocated and double cleaned polygons (the percentage is based on the whole area which has to be cleaned). The last two columns show the same values, except that they are averaged over the three simulation runs.

The table clearly shows that the percentage of double work increases when the communication range or the number of robots is reduced. If the communication radius is smaller, then the robots can not communicate that often, and if more robots are used, new information is distributed faster. Finally it can be said that even in a very

bad case ($r_{com} = 7$ and $n = 2$) the percentage of double work is very low and that in the worst case we investigated ($r_{com} = 4$ and $n = 2$) the percentage of double work is still acceptable.

5 Summary, Conclusion, and Future Work

In this paper we described a method to partition an area among multiple robots. Our approach works completely dynamic, i.e. the subareas are determined and assigned during runtime. To our knowledge, there is no other system that takes such a dynamic approach.

The method is based on a completely decentralized approach. There is neither a centralized component, nor any other global coordination needed. The robots coordinate themselves solely by regularly exchanging information.

There is no need for a global communication network. It is sufficient if the robots are able to communicate regularly. The limited communication abilities of the robots lead, in the worst case, to some additional work. The amount of additional work, however, is even under very unfavorable conditions very low. Our simulation results confirm this.

The strict separation of the area partitioning and the cleaning strategy itself makes it possible to use a lot of different cleaning strategies with our system.

As a conclusion, we can say that our approach is very suitable to partition an area among a number of cooperating cleaning robots.

For future work, we consider it interesting to examine some of the possible extensions, described in Chapter 3.6, in more detail.

We intend to use the methods described in this paper to partition an area among a fleet of real cleaning robots. We therefore currently perform some experiments with an autonomous cleaning robot (Hefter, ST82R) and with some Pioneer robots, which are all running under SINAS™ (Siemens Navigation System)[1].

References

[1] H. Endres, W. Feiten, and G. Lawitzky, Field Test of a Navigation System: Autonomous Cleaning in Supermarkets, *Int. Conf. on Robotics and Automation (ICRA)*, pp. 1779–1781, 1998

[2] S. Hert and V. Lumelsky, Polygon Area Decomposition for Multiple-Robot Workspace Division, *Special Issue of International Journal of Computational Geometry & Applications on Applied Computational Geometry*, vol. 8, no. 4, pp. 437-466, 1998

[3] H. Bast and S. Hert, The Area Partitioning Problem, *Proceedings of the 12th Canadian Conference on Computational Geometry*, pp. 163-171, Fredericton, New Brunswick, 2000

[4] M. W. Bern, S. A. Mitchell, and J. Ruppert, Linear-size nonobtuse triangulation of polygons, *Discrete & Computational Geometry*, vol. 14, pp. 411-428, 1995

[5] I. T. Christou and R. R. Meyer, Optimal Equi-partition of Rectangular Domains for Parallel Computation, *Journal of Global Optimization*, vol. 8, pp. 15-34, January, 1996

[6] I. T. Christou, Distributed Genetic Algorithms for Partitioning Uniform Grids, *University of Wisconsin Madison, Dept. of Computer Sciences Technical Report MP-TR-96-09*, Madison, Wisconsin, 1996

[7] J. Yackel, R. R. Meyer, and I. T. Christou, Minimum Perimeter Domain Assignment, *Mathematical Programming*, vol. 78, no. 2, pp. 283-303, Aug. 1997.

[8] S. Hert and B. Richards, Multiple Robot Motion Planning = Parallel Processing + Geometry, *Lecture Notes in Computer Science*, (to appear), 2001

[9] M. Schneider-Fontán and M. J. Matric, Territorial Multi-Robot Task Division, *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 815-822, 1998

[10] T. W. Min and H. K. Yin, A Decentralized Approach for Cooperative Sweeping by Multiple Mobile Robots, *International Conference on Intelligent Robots and Systems (IROS)*, pp. 380-385, 1998

[11] C. Hofner and G. Schmidt, Path Planning And Guidance Techniques For An Autonomous Mobile Cleaning Robot, *International Conference on Intelligent Robots and Systems (IROS)*, pp. 610-617, 1994

[12] E. U. Acar, Y. Zhang, H. Choset, M. Schervish, A. G. Costa, R. Melamud, D. C. Lean, and A. Graveline, Path Planning for Robotic Demining and Development of a Test Platform, *In Proceedings of the 3rd International Conference on Field and Service Robotics (FSR)*, pp. 161-168, 2001

[13] M. Jäger and B. Nebel, Decentralized Collision Avoidance, Deadlock Detection, and Deadlock Resolution for Multiple Mobile Robots, *Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1213-1219, 2001

[14] J. C. Latombe, Robot Motion Plannig, Kluwer Academic Publishers, Boston, 1991

[15] E. Chang and R. Roberts, An Improved Algorithm for Decentralized Extrema-finding in Circular Configurations of Processes, *Communications of the ACM*, vol. 22, pp. 281-283, 1979

[16] R. G. Smith, The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver, *In IEEE Transactions on Computers*, number 12 in C-29, pp. 1104-1113, 1980