Simulation of a System Architecture for Cooperative Robotic Cleaning

Hugo Costa, Pedro Tavares, Joana Santos, Vasco Rio and Armando Sousa

Abstract The increase of the use of Autonomous Vehicles in different types of environments leads to an improvement of the Localization and Navigation algorithms. The goal is to increase the levels of efficiency, security and robustness of the system, minimizing the tasks completion time.

The application of cleaning robots in domestic environments have several advantages however some improvements should be performed in order to develop a robust system. Also in large spaces one robot doesn't achieve the desired performance in terms of robustness to faults and efficiency in the cleaning process. Considering a fleet of autonomous robots, this process could be improved. The purpose of our paper is the presentation of an architecture for management a fleet of cleaning robots, considering a complete coverage path planning for large and structured environments. Compartments are found in a grid-like decomposition and an area coverage strategy are evolved (optimized) by using Genetic Algorithms. The Task allocation module is based on Auctions strategy, thus obtaining cooperation under dynamic constraints in complex environments. The case study optimizes the number of robots involved in the cooperative cleaning of a full building in the campus, based on its real architectural plans.

Keywords Multi-robot cooperation · Cleaning · Path planning · Area-covering · Genetic algorithms

H. Costa(✉) · P. Tavares · V. Rio · A. Sousa
FEUP - Faculty of Engineering, University of Porto, Porto, Portugal
e-mail: {ee10112,ee10131,ee04196}@fe.up.pt

J. Santos · A. Sousa
INESC TEC - INESC Technology and Science (formerly INESC Porto), Porto, Portugal
e-mail: {ee09133,asousa}@fe.up.pt

© Springer International Publishing Switzerland 2016

L.P. Reis et al. (eds.), *Robot 2015: Second Iberian Robotics Conference*,
Advances in Intelligent Systems and Computing 417,
DOI: 10.1007/978-3-319-27146-0_55

1 Introduction

Cleaning Robots belongs to the large group of service robots, which have the ability to perform basic housework tasks but also take care of people like a vigilant. Vacuum Cleaning Robots have gained relevance among household users. However, commercial applications typically are designed to be applied in small spaces and offices. In large environments like airports and train stations with a lot of free space, the cleaning results are better than in environments with many obstacles and compartments. Considering the structure and the dimensions of the space, a fleet of cleaning robots guarantees a more efficient cleaning in terms of tasks execution time and complete area coverage than a single robot. Some works using low-cost robotic vacuum cleaners are presented in the Literature Review. [4] presents a robust localization system considering the low-precision of the sensors used. [12] was presented a prototype multi-robot system of vacuum cleaning robots. The robot platform used was the iRobot Roomba vacuum cleaners which already have a simple algorithm for cleaning an area.

In this paper we propose a centralized architecture to manage a fleet of cleaning robots. The validation of the proposed methodology was made, using real architectural plants of a full building with many compartments. The scheduling of the tasks in a multi-robot system is a typically problem that have been studied in the State-of-Art. Several works uses a market-based mechanism [8] [7], where robots exchange their tasks over time, in order to minimize the total cost function. This framework is advantageous mainly for static environments. Similarly, in [9] each robot as the capacity of sharing part of its task with another robot, in order to an efficiently tasks allocation. Other methods to solve the Multi-Robot Task Allocation is based on Bid-Auctions, like in [6] and [10]. In [6] a dynamic bid auction finds a task allocation that reduces two parameters, the task completion time as well as the cost incurred by individual robots. The strategy used in the CleanSim is also based in auction Strategy ensuring cooperation in complex and structured environments. Cleaning Tasks require a robust area coverage method which builds a trajectory capable to cover every unoccupied area. In [5] is proposed a method based on genetic algorithms that guarantees the complete area coverage path planning. Genetic Algorithms solves many optimization problems and are based on an iterative approach inspired in the evolution via natural selection. Basically, initial solutions are randomly generated, called chromosome or genome, and then these populations are evaluated according with fitness functions. These functions includes the parameters which the system wants to minimize. Individuals with higher fitness values have higher probability of passing to following generations. Another interesting approach to solve optimization problems, are based in neural networks, like in Article [3]. The dynamics of each neuron is given by a shunting neural equation. Each robot treats the other robots as moving obstacles [11]. The application of cleaning methods based in genetic algorithms or neural networks requires that the area to be cleaning has been divided into elementary sub-areas, called Cellular Decomposition, [2] and [4].

2 System Architecture

The first step was to develop a modular architecture that allow us to increment or decrement the number of robots available and the map information. Three main systems comprises the developed framework:

- Navigation
- CTPS: Central Task Planning System
- Cleaning Method

Figure 1 presents the interactions between these modules for a multi-robot approach, considering *N* robots. CleanSim denotes the simulator implemented in python that will be exposed in Section 3.

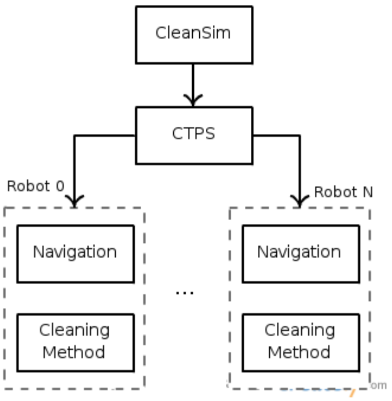

Fig. 1 General Architecture of CleanSim.

In order to implement a cooperative cleaning system was selected a centralized architecture, which is composed by a team of robots and a CTPS (Central Task Planning System). All the robots report to the CTPS and this one decides what each robot should do and in which way. These reports are not only made by the robot position and state of tasks, but also by its internal condition and the avoidance of non-predicted obstacles. This centralized system presents many advantages:

- Event log;
- Easy to change the number of robots;
- World state consultation by a robot;
- Keep the processing power away from the robots allowing their simplicity;
- Prevent the network channel from over overuse because the robots cannot communicate between them;
- Let you easily create a parallel between the simulation and a practical situation

To implement the proposed solution a data structure were carefully designed, with two major goals: modularity and expansion, as we can see in the Figure 2. The data structure is composed by two main groups, Robot and Building. The Robot is a simple model of a differential one with the capability of receiving and performing simple orders. The Building is a group of Floors, each one with Doors, Rooms and Dirtiness. All this elements are disposed in graph allowing global navigation between Rooms and Local navigation inside the Room. Each individual room has its own area and cleaning model. The robots are assigned to the rooms which should be cleaned by an action according with several parameters that will be explained in Section 2.3.

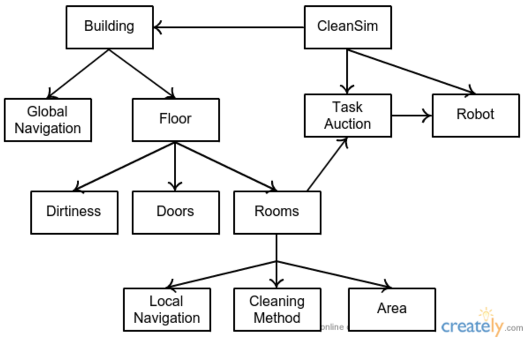

Fig. 2 Data Structure of the developed architecture.

2.1 Navigation

The navigation was divided into two sub-modules, a Local Navigation and a Global Navigation. Local navigation allows the robot navigates between two points inside the same room, being impossible to cross the door. Global navigation allows the robot to travel between rooms (door to door). This implementation simplifies a large and complex problem into two small and easier ones reducing drastically the processing time. This gain more importance when the rooms are smaller because the time to calculate the path could be significant or even overcome the time needed to clean the room.

Both approaches calculates the minimum path between two points under a graph which contains all robot's accessible points. It is only considered that the robot can travel horizontally, vertically and in the diagonal direction (45°). Dijkstra Algorithm was applied to find the shortest path, if it exists.

The higher room simulated in the Local Navigation Module have $25m \times 16m$, which generates a graph with 2520 nodes. In this case, the algorithm calculates the minimum path in $40ms$.

Global Navigation uses the same method that the Local, however in order to minimize the time complexity, in the graph building the doors isolate large group

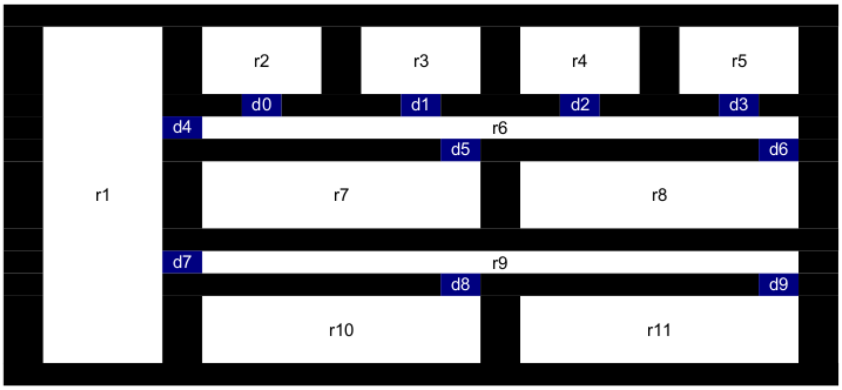

Fig. 3 Graph for the Global navigation. Notes that each room is represented as only one node.

of cells. Each room is represented with only one node in the graph of the Global Navigation, reducing the problem complexity to $n_{doors} + n_{rooms}$.

Throw the conjugation of this two navigation methods, the robot can travel between every two map points efficiently.

2.2 Central Task Planning System

At this point, the robots are already capable of traveling between rooms and cleaning them with a good strategy.

The tasks' allocation to each robot is handled by the CTPS with a dynamic auction. This approach is based on the resources' allocation according with the system needs. In our case, robots are resources and compartments are the needs. Available robots are auctioned for cleaning a compartment. A robot choose the room that placed the highest best, which is defined by Equation 1. This function is based on the following parameters:

– d_R - Distance from the robot to the bid compartment.
– d_{Rs} - Distance of the other robots to the compartment.
– t_{clean} - Cleaning Time for a compartment.

$$n_{robot} = -d_R + \frac{d_{Rs}}{N_{robots} - 1} + \frac{t_{clean}}{k} \tag{1}$$

Equation 1 guarantees the better solution considering:

– the nearest robot;
– the minimization of the bidding value even with many available robots;
– and giving higher priority to the larger rooms reducing the global time;

2.3 Cleaning Strategy

Cleaning robots should be capable of sharing spaces with humans, which mean that its trajectories should be predictable. In order to achieve this feature the cleaning area is divided into rectangular areas, called elementary areas, creating restricted spaces where the robot can navigate increasing its predictability. A Cleaning area comprises the set of nodes belonging to a compartment that should be considered by the robot during the cleaning process. An area is considered clean when the robot go through all these points. These areas are defined applying a Table Based Decomposition algorithm, exposed in Algorithm 4.

- $l[\ ], c[\ ]$: arrays with the lines and rows transitions between wall and passage.
- A_{start}: initial starting area with l rows and c columns.

```
1: $[l[\ ], c[\ ]] = FindTransition();$
2: $A_{start} = AreaDefinition(l[\ ], c[\ ]);$
   {$a_{start}$ have $n_l + 1 \times n_c + 1$ areas. Each resultant areas are represented by one pixel called cell.}
3: **loop**
4:    $IncreaseNumberCell();$
      {Increase the number of cells}
5:    **if** $GroupCells()$ **then**
6:       **return**
         {Try to group in every way each cell until it is not possible to group any more.}
7:    **end if**
8:    $GenerateNewAreas();$
      {Obtain new areas of each cell.}
9: **end loop**
```

Fig. 4 Table Based Decomposition Algorithm

Algorithm 4 divides the map as a set of elementary areas to be cleaned. The next step was to find the best order which these areas should be cleaned as well as the optimal/sub-optimal path to navigates and clean each area. Genetic Algorithms were used in order to find a near optimal solution that minimizes the cost function. In our framework the implementation of Genetic Algorithms was advantageous mainly if we consider the total number of possible combinations.

Considering a compartment with n areas, the number of cleaning order possibilities is given by:

$$C_{order} = !n; \tag{2}$$

Each area can be cleaned using 16 different methods (see Table 1), so the number of possibilities to clean an area is:

$$C_{path} = 16^n; \tag{3}$$

Table 1 Combinations of cleaning methods for each area

		gene[0]	gene[1]	gene[2]	gene[3]
0000	0	circular	anti clockwise	top	right
0001	1	circular	anti clockwise	top	left
0010	2	circular	anti clockwise	bot	right
0011	3	circular	anti clockwise	bot	left
0100	4	circular	clockwise	top	right
0101	5	circular	clockwise	top	left
0110	6	circular	clockwise	bot	right
0111	7	circular	clockwise	bot	left
1000	8	S	y Axis	top	right
1001	9	S	y Axis	top	left
1010	10	S	y Axis	bot	right
1011	11	S	y Axis	bot	left
1100	12	S	x Axis	top	right
1101	13	S	x Axis	top	left
1110	14	S	x Axis	bot	right
1111	15	S	x Axis	bot	left

The combination of Equations 2 and 3 gives the total number of possibilities:

$$C = C_{order} \times C_{path} = n! \times 16^n \tag{4}$$

A genetic algorithm comprises 4 layers which are genes, chromosomes, individuals and population. In our approach, each compartment is an individual to which should be assigned a cleaning method. Always that a robot navigates to a compartment, need to know how the better way to clean that. This information is sent by the CTPS module with a chromosome. A chromosome is constituted by n sets of 4 genes ($cleanMethod()$), according with the Table 1. Each gene represents the order which the areas should be cleaned ($sortGene()$).

– **CleanMethod():** Cleaning Method for each area. This is composed of 4 boolean genes where a mutation corresponds to a value inversion.
– **SortGene():** Sort a gene with the order which the areas of a compartment should be cleaned. It's an array between $[0, n-1]$, where n represents the number of areas of a compartment. Each gene mutation has a probability of occurrence.

An initial solution was considered in the $SortGene()$ function in order to accelerate the convergence of the algorithm. This initial solution was generated using a greedy solution. Besides the solution found wasn't the optimal solution, in our approach a sub-optimal solution is enough.

3 Simulator

In order to a better validation and evaluation of our framework, it was created a simulator, called CleanSim.

3.1 *Environment Acquisition*

In order to keep the modularity and adaptability of the system, the map is obtained by processing CAD files from the building. The processing algorithm occurs in three steps, first, if needed, the user edits the file to add or to change necessary information. Then, all the unnecessary layers are removed and the image is converted to bitmap. In this format the doors are identified and painted as blues, all the cleaning areas should be white and the rest in a different color. Also a dirtiness layer is created, where the pixel intensity represents the amount of dirt. Secondly, the image is imported to the simulator and scanned to identify all the objects, door and rooms, through a labeling technique with connectivity-8. Then, among all the identified objects, the system searches for the objects that are huddled together creating a graph that will be used to create paths. The results from this process are saved into hard-drive allowing the program to import these results instead of calculate them every time it is switched on.

3.2 *Dirtiness Simulation*

In order to simulates the dirtiness, two images were overlaid to the map, as can be seen in the Figure 5.

The first image (shown in red) represents the variation of the dirtiness, according with Equation 5. $S_{x,y}$ denotes the dirtiness in each pixel. $P_{x,y}$ gives the derivative of the function $S_{x,y}$ in order to the time. The second layer is the dirtiness of each pixel on the initial time instant.

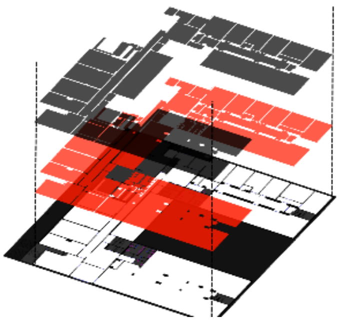

Fig. 5 Representation of the layers used to the dirtiness simulation.

$$P_{x,y} = k \times \frac{dS_{x,y}}{dt} \tag{5}$$

4 Results

Some experiments were performed at different architecture levels. Firstly, will be presented the navigation results and the advantages of the proposed modular architecture. Then, genetic algorithms for area covering and the task allocation system will be detailed.

4.1 Navigation

The path planning was separated in global navigation and local navigation, both of them based on a navigation grid. This approach splits a complex problem in to small ones, allowing the system to be scaled to more complex facilities including elevators and several buildings. Also, with this implementation it is possible to use different path planning algorithms for different robots and aggregate several parts of the building if necessary. In this way, the navigation can be adapted to the global and local needs.

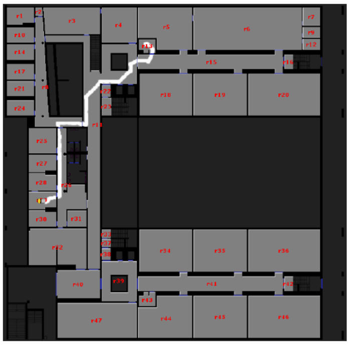

Fig. 6 Navigation between two rooms.

In Figure 6 is represented the path produced by the Navigation system. Notes that the grid resolution and the fact that the robot can only moves at multiple angles of 45° limits the solution. Comparing the presented result with the optimal one, if we have an infinitesimal grid, the total length of the trajectory would pass from 32.2m to 30.7m representing a gain of 5%.

4.2 Area Covering

Genetic Algorithms were used to the optimization of the path made by the robot in order to completely clean a room. Genetic evolution is a method that after a few iterations can present a good solution to the problem. It is necessary to find an advantageous number of iterations to stop the algorithm evolution. The strategy is based on an evolution of 4^n times a population of 20 individuals, measuring the time variations. Each population of 20 individuals represent current solutions, and the combinations of them allows to find more solutions which will be future generations.

Table 2 Genetic Evolution on a Case Study Generic Floor

Generations	1	4	16	64	256	1024
Individuals	20	80	360	1280	5120	20480
Time(s)	10294	10213	10103	9978	9874	9860
$\frac{\Delta T}{n_{generations}}(s)$	0	27.167	9.167	2.609	0.542	0.018

As presented in Table 2, the first generation gets a time variation of $27s$ approximately and in the last generation this time is reduced to $0.018s$. Predicting the same behavior, it is expectable that in the next 1024 generations the total time variation would be around $0.47s$. In the first 1024 generations we obtain a reduction of time in $434s$ and by duplicating them the gain is only 0.11%. It's possible to conclude that for this practical case, 1024 generations are enough.

4.3 Task Allocation

To allocate a task to a robot we propose an auction method, where the rooms bid an available robot. The robot is attributed to the winner. Biding value is a combination

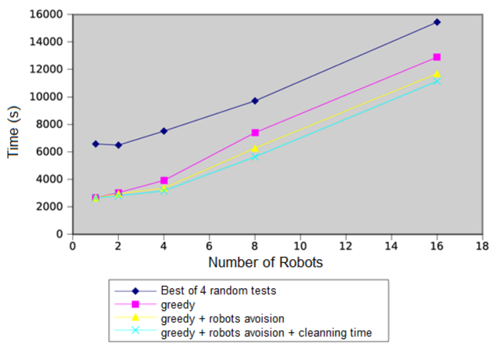

Fig. 7 Comparative performance of different task allocation methodologies.

of three factors: proximity to the robot (greedy), distance of other robots (robots avoidance) to the room and cleaning time. Several simulations were performed with different number of robots and biding strategies.

Figure 7 represents the total unproductive time taken by the robots cleaning a building. The active cleaning time is constantly $9857s$. As we can observe the non-productive time is always above $2000s$ increasing with the number of robots. This time is given by the traveling time between the starting positions until the room that should be cleaned. Also the firsts robots to finish, have to wait for the last.

5 Conclusion

The purpose of this work is the presentation of a modular architecture which manages a fleet of cleaning robots. Three main systems were developed: Navigation Module (which was divided into local and global), the Central Task Planning and a Cleaning Technique that uses genetic algorithms to improve the efficiency of the cleaning method. All modules were implemented in a centralized processing unit that is aware of everything and capable of bidirectional communication with each individual robot. In order to make the system suitable for different building, the map is not static, being directly imported from a CAD file with minor modifications. The main contributions of this project are the addition of genetic approaches to solve the cleaning problem and the usage of an auction task allocation and division in a dynamic way. The simulation experiments were performed using a developed simulator, called $CleanSim$, which includes the simulation of the map dirtiness. As future work, the CleanSim environment should be improved, adding random factors like peoples, obstacles and doors. These elements will allow to create more realistic simulations.

Acknowledgments This work is partially financed by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project ≪FCOMP-01-0124-FEDER-037281≫.

References

1. Luo, C., Yang, S.X.: A real-time cooperative sweeping strategy for multiple cleaning robots. In: IEEE International Symposium on Intelligent Control (2002)
2. Oh, J.S., Choi, Y.H., Park, J.B., Zheng, Y.F.: Complete coverage navigation of cleaning robots using triangular-cell-based map. IEEE Transactions on Industrial Electronics (2004)
3. Luo, C., Yang, S.X., Stacey, D.A.: Real-time path planning with deadlock avoidance of multiple cleaning robots. In: IEEE International Conference on Robotics and Automation (2003)
4. Pinheiro, P., Cardozo, E., Wainer, J., Rohmer, E.: Cleaning Task Planning for an Autonomous Robot in Indoor Places with Multiples Rooms. International Journal of Machine Learning and Computing (2015)

5. Jimenez, P.A., Shirinzadeh, B., Nicholson, A., Alici, G.: Optimal area covering using genetic algorithms. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics (2007)
6. Dasgupta, R.: A Dynamic-bid Auction Algorithm for Cooperative, Distributed Multi-Robot Task. Technical Report, Department of Computer Science University of Nebraska at Omaha (2009)
7. Dias, M., Anthony, S.: A free market architecture for distributed control of a multirobot system. In: 6th International Conference on Intelligent Autonomous Systems (2000)
8. Dias, M., Anthony, S.: Opportunistic optimization for market-based multirobot control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2002)
9. Robert, Z., Stentz, A.: Complex task allocation for multiple robots. In: IEEE International Conference on Robotics and Automation (2005)
10. Robert, Z.: An Auction-Based Approach to Complex Task Allocation for Multirobot Teams. PhD Thesis, The Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania
11. Santos, J., Costa, P., Rocha, L.F., Moreira, A.P., Veiga, G.: Time enhanced A*: towards to the development of a new approach for multi-robot coordination. In: IEEE International Conference on Industrial Technology (ICIT) (2015)
12. Nikitenko, A., Grundspenkis, J., Liekna, A., Ekmanis, M., Kulikovskis, G., Andersone, I.: Multi-robot system for vacuum cleaning domain. In: 12th International Conference, PAAMS, Salamanca, Spain (2014)