A Multi-Robotic System for Environmental Dirt Cleaning

Chuong P. Le¹, Anh Q. Pham², Hung M. La³, David Feil-Seifer³

Abstract—An industrial environment usually has a lot of waste that could cause harmful effects to both the products and the workers resulting in product defects, itchy eyes or chronic obstructive pulmonary disease, etc. While automatic cleaning robots could be used, the environment is often too large for one robot to clean alone in addition to the fact that it does not have adequate stored dirt capacity. We present a multi-robotic dirt cleaning algorithm for coordinating multiple iRobot-Creates as a team to efficiently clean an environment. Often, since some spaces in the environment are clean while others are dirty, our multi-robotic system possesses a path planning algorithm to allow the robot team to clean efficiently by increasing vacuum motor power on the area with higher dirt level. Overall, our multi-robotic system outperforms the single robot system in time efficiency while having almost the same total battery usage and cleaning efficiency result. The project source codes is available on our ARA lab's github: https://github.com/aralab-unr/multi-robot-cleaning

I. INTRODUCTION

In an industrial environment, such as a factory or a warehouse, the work conditions are harsher than normal with wastes - hazardous or non-hazardous lying around, which the workers are exposed to every day. Since hazardous wastes cause visible damages, they are often more carefully handled than non-hazardous ones like metal dusts or ashes, the damages of which are long-term and not always visible. One of the damages that non-hazardous wastes can cause is product defects, sabotaging the purpose of the product and making the producer look bad. In addition, workers working in such an environment may have chronic medical issues such as skin problems, sinusitis, and eye problems. In particular, if inhaled, the dust may cause such symptoms as coughing or breathing issues and, in more serious cases, could damage the worker’s lungs or other organs [1]. If they try to clean the environment by sweeping the dust, it will go into the air and create a higher chance of the workers inhaling it.

In this situation, using a dirt cleaning robot, such as the iRobot Roomba, would be a logical solution. Commercial automatic cleaning robots have become very popular nowadays because of their ability to clean autonomously. However, one of the earliest problems with it is that their cleaning process is completely random, and there is no guarantee that they would be able to clean all the space. Recently, though, there are new models of iRobot that could map an entire environment and systematically clean it.

Even though mapping and cleaning every inch of the environment indiscriminately can be an effective method, it lacks the ability to plan the best path as some areas of the map might be dirtier while others remain clean. Having a dirt model to estimate the amount of dirt in each small area of the environment will help create a more efficient path of cleaning for the robot to follow and, thereby, will increase cleaning efficiency.

Although most automatic cleaning robots could systematically clean a whole environment, an industrial environment is much too large for one robot to clean by itself in addition to the fact that the amount of dirt will exceed the dirt stored capacity of one robot. In this project, we will be using multiple iRobot Create 2 because of its low cost and its programmable platform [2] to assemble a multi-robotic cleaning team. In addition, we used the hokuyo laser scanner, and the dirt sensor that is already equipped in the iRobot Create 2 to create a dirt map of the environment. Then we implement the multi-robotic system, where all robots share the same map, know their positions in it, and effectively communicate with each other, to clean an entire industrial environment efficiently.

This paper is presented in the following order: After the related work, we will be discussing the method of
mapping and localizing with the Simultaneous Localization and Mapping (SLAM) and the Monte Carlo Localization system. After that, we will do a brief overview of the cell-wise Poisson process by Hess et al. [3] that we will be using to build a dirt map. Then, we will present the algorithms for our multi-robotic system to create an efficient cleaning path for each robot in the team. Finally, we will present the result of our experiment, evaluate our work and discuss the future usage of our results in the conclusion.

II. RELATED WORK

Multi-robotic system has many benefits, mainly because it can accomplish task that single robot could not or take much longer time to do. Prime examples of its benefits include [4]–[15]. [9] uses rapidly exploring random tree (RRT) for vacuum robot navigation and path planning for large indoor vacuum cleaning. [7] by Connell and La uses the RRT\(^2\) for path replanning in a non-static environment. The multi-robot algorithm here yields equivalent or better paths and planning time efficient. [11] by Pham et al. creates a wildfire distribution model using multiple unmanned aerial vehicles (UAVs) to track and predict wildfire spreading. In this case if they were using a single UAVs it would not be able to cover the entire fire and create the wildfire model. Another case of using multiple robots for scalar field mapping [16], [17] that clearly indicated the benefit of multi-robot teaming rather than a single one.

If the robot is cleaning the same environment again, it would be a waste of time to re-scan the environment. In a paper by Zaman et al. [18], it was shown that laser and odometry data from scanning of a whole environment using SLAM can be saved into Yet Another Markup Language (YAML) and portable gray-map format (PGM) files using map saver and map server. YAML files shows the image while PGM shows the descriptions of the map. When implementing the Adaptive Monte Carlos Localization (AMCL) with the YAML and PGM file, the robot can localize itself inside the saved map [19].

One of the work used to navigate efficient path planning and navigation is the cell-wise Poisson process that used equations from the homogeneous Poisson process and the maximum likelihood estimator to predict where the dirt will be after allowing the robot to clean through the environment several times. Then it would create a dirt map that predicts the intensity of dirt in each cell and, using the Traveling Salesman Problem (TSP), determine an efficient path for the iRobot to clean. The project’s benefits include less noise and less energy consumption in the robot, because of the Adaptive Poisson cell-wise method. In this paper, we will create a dirt map and apply the multi-robotic system and propose path planning algorithm for multiple robots to clean the environment in an efficient manner.

III. METHOD

As mentioned before, we used 2 iRobot Create. We used the iRobot Create due to the fact that it is an affordable platform and has a built-in dirt detection sensor at the bottom inside the suction unit. The iRobot Create uses a piezoelectric sensor, which generates electrical pulses when dirt hits it and gives a measurement of the dirt reading [20]. We also mounted a hokuyo laser scanner. In the experiment, we used cardboard boxes to create an environment for the iRobot Create to map and clean and substituted play sand as dirt. The play sand were scattered randomly in the amount of 10 grams per cycle.

As mentioned before, we used 2 iRobot Create. We used the iRobot Create due to the fact that it is an affordable platform and has a built-in dirt detection sensor at the bottom inside the suction unit. The iRobot Create uses a piezoelectric sensor, which generates electrical pulses when dirt hits it and gives a measurement of the dirt reading [20]. We also mounted a hokuyo laser scanner. In the experiment, we used cardboard boxes to create an environment for the iRobot Create to map and clean and substituted play sand as dirt. The play sand were scattered randomly in the amount of 10 grams per cycle.

\[
E[N^c(t)] = \frac{(t - s)}{\sum(t_i = t_{i-1})} \sum_{i=1}^{n} k_i^c,
\]  

where \(s\) is the latest cleaning. Every time the iRobot past through each cell \(c\), it takes the dirt reading \(k_i^c\) cleans the cell, and saves the time \(t_i\). Therefore, \(\sum_{i=1}^{n} k_i^c\) stands for the sum of all dirt reading of cell \(c\).
C. Multi-robotic Cleaning System

Our multi-robotic cleaning system utilizes two nodes function - Goals and send - as seen in Figure 2. The Goals node uses Algorithm 1 and 3 to divide the environment into separate cleaning spaces. Then the send node will use Algorithm 2 and 4 to find the best cleaning path for each iRobot. Create and send them there.

1) Multi-Robot Networking: To connect the iRobots together, we connected all machines to the same router through local area network and implemented the MultiMachine in Robotic Operating System (ROS) to configure multiple machines to use a single master using hostnames and IP addresses of the Intel NUCs on each iRobot. The master machine will be a laptop that we used to receive data from the iRobots and send directions to it. Figure 3 provides a visualization of our multi-robot networking.

2) Multi-robot Divide: After completing the dirt map, we need to divide it into separate cleaning spaces for each iRobot. To make the cleaning time of each iRobot relatively close, we tried to make the grids have even (or close to even) \( \lambda \). Our grid divided by Algorithm 1 uses a recursive method that divides the environment into many grids. The algorithm calculates the dirt level of each grid \( \lambda_g \) by \( \lambda_{\text{total}}/\text{number of robot} \). It starts with grid1 being the original grid map with all available cells of the dirt map while the other grids have all unavailable cells. The recursive function would, as it finds the cells grids to divide, transfer the available cells from grid1 to grid2 until \( \lambda_g \) is reach, then it will move on to the next grid to transfer until the number of grids divided gridDivided is equal to the number of grids \( \text{num} \). Since most of the time in real life, the grid’s \( \lambda \) can not be evenly divided, we need to make sure that the grids we can make them close to even so we put \( p \) as the counter that increases if the map can not be divided evenly. For example, if a map can not be divided into 5 and 5, \( p \) would increment so that map will try 4 and 6. Even though we are using just two vacuum robots, any number of robots, \( \text{num} \) can be implemented in this algorithm because it will divide the environment into \( \text{num} \) cleaning spaces.

### Algorithm 1: Grid Divide Algorithm

1. **Input:** number of robots \( \text{num} \), total dirt level \( \lambda_{\text{total}} \), grid \( V \)
2. \( \text{gridDivided} = V \) set all cells \( c \) in other grid2-num to 0
3. \( p \) is a counter increase if map can’t divide evenly
4. gridDivided is the counter dividing grids
5. **while** gridDivided < num **do**
   6. **while** \( p < \lambda_{\text{total}}/\text{num} \) **do**
      7. find the starting cell \( c_{\text{start}} \) using row major order
      8. \( l = \lambda_{\text{total}}/\text{num} - p \)
      9. \( \text{recursive}(c_{\text{start}}, l) \)
      10. **if** solution is found **then**
          11. break
      12. **else**
          13. reset the 2 grids
          14. increment \( p \)
      15. **end**
   16. **end**
   17. increment gridDivided
18. **end**

Fig. 4. An example of the path planning algorithm we create for each iRobot in their respective cleaning space. The cell colors represent \( c_j \). The deeper the color, the higher the dirt level and motor power. After the iRobot cleans a cell, it will be labeled as clean/white as shown on the right picture with the red outlined cells.

3) Path Planning: After assigning each iRobot to their respective cleaning grid map, we created a path planning algorithm (Algorithm 4) that utilize the A* Search Algorithm as presented in Algorithm 2 for each for each robot. We used the A* because of its ability to find the shortest path possible as well as the fact that it also includes obstacle avoidance [23].

Our A* algorithm finding path by searching for the nearest cell \( c \) and using the heuristic function. It ignores the cells that are already cleaned \( c_3 \) and obstacles. In addition, if it finds a viable diagonal cell to go to, if there is obstacle on either side of the two cells, it ignores that cell as well.

In our path planning algorithm, we want to prioritize the cell with the highest \( \lambda \) before the clean cell because...
Algorithm 2: A* Path planning Algorithm

1 Initialize closedList and openList
2 Note: closedList and openList is a list of cells \( c \) each has: coordinate \((x, y)\), cell dirt level \( \lambda \), distance from starting cell to the next cell \( g \), distance from next cell to goal \( h \), parameter of \( g + h \) \( f \)
3 Put the starting cell \( c \) in openList
4 while openList is not empty do
5 find the node with the least \( f \) in the openList \( q \) and pop \( q \) off openList
6 find \( q \)'s 8 cell path \( p \) (east northeast north east-west west southwest south southeast) and set their parents to \( q \)
7 for each \( p \) do
8 if \( p = \text{goal} \) then
9 stop searching and break out
10 end
11 \( p_h = \) distance from \( p \) to \( \text{goal} \), \( p_g = \) distance from \( q \) to \( p \) + \( q_g \), \( p_f = p_h + p_g \)
12 if a cell \( i \) has the same position as \( p \) in openList then
13 if \( i_f \) less than \( p_f \) then
14 skip it
15 end
16 end
17 else if a cell \( i \) with the same position as \( p \) in closedList then
18 if \( i_f \) less than \( p_f \) then
19 skip it
20 end
21 end
22 else if \( p \) is cleaned then
23 skip it
24 end
25 else if path to \( p \) is diagonal then
26 if there is obstacle(s) on either side then
27 skip it
28 end
29 end
30 else
31 add the node to the openList
32 end
33 end
34 push \( q \) in closeList
35 end

if the machine breaks down, we have already cleaned the dirtiest part of the map. In addition, we want to use more vacuuming force when cleaning dirtier cells. As can be seen in Algorithm 2, it finds and clean the cell with the highest \( \lambda \) before the counter decrements to the next dirtiest cell. However, the problem is that if we use this method to go to every single cell, it would take a very long time. Our solution to that is on the way it should also cleaned the cells that it passed and label it as cleaned so the A* algorithm can ignore that cell while finding a path. The iRobot should use different motor force depending on \( \lambda \). In a real situations, there may be times when there will not be a path that ignores all of the cleaned cells. If that is the case, then just for that path, we use an alternate A* Search Algorithm that does not ignore cleaned cells. That algorithm will just be Algorithm 2 without line 26-28.

Our multi-robotic system is demonstrated in figure 6 where it divided the environment into two cleaning space for two iRobot to path plan and clean.

IV. EXPERIMENT
A. Testing multi-robotic system versus the built-in system

The experiment setup is shown in Figure 1.

In our experiment, we are testing our multi-robotic system against the iRobot Create’s cleaning system and evaluate the performance of cleaning time, total battery usage. First, we timed the iRobot Create’s cleaning process until it is finished and returned it its home-base. Then we check the battery usage percentage and recorded it. Then, we implemented our multi-robotic system and timed its cleaning process. When all iRobots stop, we also measure each of its battery usage percentage using the information display from [24], added
Algorithm 3: Recursive Function

1 *Note: (for ALL Algorithms in this paper) even though the cell dirt level is [0,77], in the algorithm we made it so that λ: [0,3], where 0 = [0,19], 1 = (19,38], 2 = (38,57], and 3 = (57,77].
2 Function recursion(starting cell c, grid’s required λtotal)
3 if λtotal is 0 then
4     if the grid1 is connected and grid2 is connected then
5         solution is found
6     end
7     return;
8 end
9 set starting cell’s position in grid1 to unavailable
10 set starting cell’s position in grid2 to available
11 λtotal -= cλ
12 if λtotal ¡ 0 then
13     return
14 end
15 find neighboring cells of east, west, south, north
16 for each neighboring cell do
17     if cell is available then
18         recursion(cell, λtotal)
19     end
20 end
21 return

Algorithm 4: Send

1 Initialize V an cell array Initialize levelCounter the counter that keeps track of the current highest λ
2 Set levelCounter to highest λ of a single cell
3 while levelCounter bigger than 0 do
4     find all cell with λ = levelCounter and put it in V
5     while V is not empty do
6         find the cell cleast with the least distance away from starting point find the path to cleast using the A* Path Planning Algorithm
7         if path is not found then
8             use an alternative A* star path planning that does not ignore clean cells
9         end
10     go to cleast along the way, turn the dirt motor power on according to the dirt level of the cell on the way and change its status to cleaned/λ = 0 remove cleast from V
11 end
12 decrement levelCounter
end

all three together and recorded it. We repeat the process ten times and took the average time and total battery usage. Overall, there were no outliers in either categories - duration and total battery usage - and the spread was relatively small. For cleaning efficiency, each time it finished cleaning, we decided to observe the environment and decided that the result is very satisfactory. The demonstration of two robots cleaning an environment can be seen in this link: https://youtu.be/sqrj5DN1SGQ

B. Result

Figure 5 presents the overall result of our experiment. While the total battery usage of the multi-robot system is higher than the built-in cleaning system of a single iRobot, the total cleaning duration of our multi-robotic system is less than 1/2 the duration of a single robot. This experiment shows the time cleaning efficiency of our multi-robotic system while having almost the same battery usage and yielding little to no dirt remaining in the environment.
V. CONCLUSION

In this paper, we present a multi-robotic algorithm along with a dirt model for industrial environment cleaning. After collecting dirt reading data, the dirt model predicts the dirt level for each small area on the environment map. Our algorithm divides the model relatively even in terms of total dirt level among the robots and creates the best path for each robot as well as cleaning duration of each small area depending on its dirt level. The proposed algorithm was implemented and validated on two iRobot-Creatives. The result was that while the battery usage of our multi-robotic system was close, its cleaning duration was approximately twice as fast as the single iRobot cleaning system. In the future, we plan to make our system viable for non-static environment such as when the map changes as well as collecting more data to update the new areas in the dirt model. In addition, we aim to have each of our iRobot be able to do replanning the path in case of a moving obstacle like [7], [25] and avoiding trapped by convex shaped obstacles by using a combination of rotational force field [26], [27] and repulsive artificial potential force field [28].

REFERENCES


Fig. 6. The motion planning of two robots with respect to the dirt level a divided environment.