Technical Proposal:Area coverage using Swarm Robotics

Shreenaga Tejas,Shubham Mittal,Sachin Badang,Akshat,Dishay Mehta

December 17, 2021

1 Motivation

The Problem Statement of AIITRA is to build and program the autonomous agents from scratch to coordinate and cover maximum amount of area in minimum time.The ideas used for the development of the algorithm for the robot is inspired by the way ants use specific smell(pheromone) for navigating and transmitting information through the colony intelligently.While the problem statement is aimed for cleaning robots it finds uses in many fields such as planet exploration,search and rescue etc.Being an NP problem there is no one correct approach for covering the whole area we present our approach in this proposal.We have verified our idea by simulating it in Gazebo by using ROS Melodic in Ubuntu 18.04.

2 Design

The robot is a 4 wheel planar drive robot with mecanum wheels. The main body of the robot is divided into three compartments. The lower most compartment houses the electronic components. The middle compartment is filled with the cleaning liquid and the upper component. A Lidar is placed on the top to aid in obstacle avoidance and locomotion.

2.0.1 Lower Component

The lower component contains most of the electric components: Rechargeable battery

- A4988 stepper motor driver
- Stepper Motor Driver Controller Board for Arduino
- Arduino UNO as control unit
- 4 ultrasonic sensors are attached to the bottom to detect the type of surface the robot is cleaning.

2.0.2 Middle Component

The middle component consists of the

- Pneumatic system: This system controls the movement of the rotating sponge. The working is analogous to the working of a syringe. Some surfaces may not be suitable for wet cleaning (Eg carpets). The ultrasonic sensor on detecting the surface sends an input to the pneumatic system via the Arduino which pulls up the sponge. Dispenser system: At regular intervals of time the dispenser dispenses the cleaning liquid
- Dispenser system: At regular intervals of time the dispenser dispenses the cleaning liquid
- Cleaning Liquid

2.0.3 Upper Component

The upper component houses the vacuum cleaning motor.The dry waste is collected in this component. The rectangular slits have one way valves attached as well to prevent the dry waste going back. The upper component also contains the air compressor for the pneumatic system.

2.0.4 Cleaning Mechanism

The robot allows both wet and dry cleaning options.

- Dry Cleaning Mechanism: The dry cleaning is done by 4 vacuum dry-cleaning slits near each edge of the bot. The 4 rubber rollers pick up the dry waste and the vacuum cleaner motor provides the necessary pressure difference and power to pick up the dry waste. The symmetric placement of the slits has a number of advantages. It makes the cleaning of the surface independent of the orientation of the bot thus reducing the time required to rotate to a particular orientation. The placement of the slits also allows simultaneous drying of the surface after the wet-cleaning.
- Wet Cleaning Mechanism: The wet cleaning mechanism consists of a rotating sponge attached to a plastic body which is connected to a motor. This motor sitting on the rubber body rotates the sponge. The pneumatic design allows the rubber to move or down depending on if the surface requires wet cleaning. Another advantage of the pneumatic design is it allows cleaning on uneven surfaces as the pressure maintained in the chamber allows it to act analogous to a spring. Pipes are attached from the liquid compartment to the sponge. Liquid is dispensed at regular intervals.

2.0.5 Docking System

This is a unique feature provided along with our cleaning bot. The docking system allows the robot to autonomously locate the docking system and dock itself when the resources are depleted. Here the batteries are recharged. The cleaning liquid is reloaded and the dry waste is taken out and stored into bags. The sponge will also be thoroughly cleaned.

2.0.6 UI and Scheduling

The cleaning bot will also come with an application to see the cleaning process in real time. The application will also provide the feature to schedule the cleaning process and the frequency of cleaning.

2.0.7 Advantages over other traditional cleaning bots

This robot has certain advantages over other traditional cleaning robots.

- The 4 mecanum wheels allow the bot to move independent of its orientation.
- The 4 vacuum dry-cleaning slits adds an extra layer of symmetricity to the bot and also allows faster drying.
- The application provides a real time interface to check the progress and to schedule the cleaning.
- The Docking system allows the bot to be truly autonomous by allowing hands free recharging, reloading and storage of dry waste.

2.0.8 Products used

S.No	TITLE	QUANTITY	COST PER PIECE (in Rs.)	NET AMOUNT (in Rs.)
1	Lidar	x1	₹ 6,499.00	₹6,499.00
2	Air compressor	x1	₹ 319.00	₹ 319.00
3	Vacuum motor	x4	₹ 400.00	₹ 1,600.00
4	Pneumatic cylinder	x4	₹ 286.00	₹ 1,144.00
5.	Arduino uno	x1	₹ 399.00	₹ 399.00
6.	Stepper motor	x4	₹ 575.00	₹ 2,300.00
7.	Motor driver	x4	₹ 74.00	₹ 296.00
8.	Bo motor	x1	₹ 55.00	₹ 55.00
9.	Light motor	x2	₹ 99.00	₹ 198.00
10.	Mecanum wheels	x4	₹ 2,933	₹ 2,933
11.	Rechargeable Battery	x1	₹ 1,689.00	₹ 1,689.00

Figure 1: Products used

3 Software

3.1 Ideas

3.1.1 Area division

For implementing this algorithm we divide the map into 1*1 cells that when the bot visits it will clean.Moreover a threshhold region is defined so that whenever it comes in that it is allowed to take a decision otherwise before visiting the cell it will go to another cell and hence reduce the cleaning area covered

This division eases the decision making process as well as breaks a countinous search space into discrete blocks.

The cell size is dependent on the robot size and the lidar range taken and can be changed accordingly.

3.1.2 Information Flow

There are two methods for flow of information between agents

- Using a central Server for communicating Map information between agents at all times
- Using a Local server on each robot and using a Map merging algorithm when robots come in the communication range of each other

The method that will be used is decided based on the use case of the agents i.e. if they are deployed in an environment where dimensions of the area is less than the communication range we will use the first method in all other cases we will use the second method.

As discussed above using of a central server means that in every iteration of the algorithm the robots communicates its visited cells and its pheromone values with the central server and the robots also receive information from the central server on what are the locations of the other robots on the unexplored map.

The second method involves using a Map merging algorithm every-time it is in communicating range of neighbouring bots. Here the neighbouring bots transmits velocity and location.(Here the rejection vector will be calculated by extrapolating the last seen location + last seen velocity*time).

3.1.3 Use of Pheromones

In this approach presented we use attractive pheromones and repulsive pheromones. Pheromone(in the context of this algorithm) are values given to the grid cells when a agent visits a cell or when it detects an obstacle.

The above approach associates pheromone values based on the number of exit cell locations available to the bot and whether it has visited the cell previously or not.

We use repelling pheromones when the agent has no other option to visit except the exit location which it has already visited and is also given when there is only exit location possible.

For example when there are more than one exit location possible we give positive pheromone value(which is equal to number of valid unexplored exit locations)so that when a different agent is nearby the cell it will visit and move in a different direction than the previous bot location and thereby cover more area.

When the bot has no exit location in unexplored area then it follows the gradient trail(the rule is to move to the highest gradient cell).This strategy allows it to move towards paths which are unexplored easily

3.1.4 Dispersion using Coloumb's Law

We think of agents as charges and generate repulsive forces between them and allows us to compute a rejection vector that decides the direction the agent is headed to.The main reason behind using Coloumb's law is that the agents do not come in close proximity of each other.This allows more area to be covered in lesser time.

3.1.5 Obstacle avoidance Using Lidar

Lidar is used for obstacle detection while we have used 0.8m ranged lidar and 54 rays and a robot of size 0.35*0.35*0.5m,depending on the use case we can change its paramaters.

- RayRange:The RayRange should be such that it detects an obstacle in a neighbouring grid cell effiecitnltly.
- NumberOfRays: it should be chosen as so that when bot is at small distance to obstacle ,it should be detected by the agent and thereby avoid it(more the no of rays more will mean accuracy).
- FieldOfView:For the algorithm to work we need to have a 360 degree lidar mounted so have a view of the eight exit location in real time.

During motion of the bot the selected no of rays return bool values whether it has hit an obstacle or not. Using this information it checks whether it has already given pheromone to it or not,if not it will give and then this grid location cannot be appended in final exit locations.If obstacle is not detected then control will be given to the algorithm.

4 PseudoCode

Inputs : Agent's current location
Outputs:Agent's movement to next exit location

def RejectionVector(LocationsOfBotswrtItself):
ArrOfVectors=Calculate(r12,r13,r14);
R12,R13,R14=CalculateUnitPerpVectorsInFirstOrSEcondQuadrant(r12,r13,r14);
W12,W13,W14=CalculateMagnitude(r12,r13,r14);//calculated using inverse square function with appropriate threshold
ResultantVector=(W12*R12)+(W13*R13)+(W14*R14);
Return ResultantVector;
def ClosestExitLocation(vector,ArrOfPossibleVectors):
theta=CalculateAnglewrtx-axis(vector);
angleArray=FindAnglewrtx-axis(ArrOfPossibleVectors);
ExitLocation=CalculateClosestAngle(theta,angleArray); //If this outputs two values it will return nothing
Begin
neighbAgents=FindNeighborAgentsPosition();
validExitNo's=FindValidExitsFromCurrentPosition();//using LIDAR detection
n=ValidExitLocationsWhichAreUnExplored()//Subscribe to centeral server or by his own map if merging system is used
if (n):
SwitchCase n:
case:0
pheromone(-2,presentGrid);
go back to previous grid();
case:1
PlaceRepellentPheromone+=1*E(SmallNegativeValue)
GoTothatExitLocation();
Default
rejectionVtr=rejectionVtr();
if(ClosestExitLocation(rejectionVtr,unexploredExitCellsVtr-currntBotVtr) or check consecutive 10 cells Unfilled pheromone // priority to more unexplored cells):
Pheromone(x,y)+=n;
Pheromone(x,y)+=1*E(smallNegativeValue);
elif(ABOVE CONDITION FAILS):
randomPath(i,i+1,i+2,....)
Pheromone(x,y)+=n;
Pheromone(x,y)+=1*E(smallNegativeValue);
End Switchcase
Else:
If(validExitNo's):
FollowGradientTrail();[Go to the cell which has maximum gradient for two cells w.r.t current cell]
Pheromone(x,y)+=1*E(smallNegativeValue);
If(ABOVE CONDITION TIE):
randomPath(i,i+1,i+2)
Pheromone(x,y)+=1*E(smallNegativeValue);

Else:
AccessCentralServer();
GoBackToPreviousLocation();
Pheromone(-2);

5 Training and Testing

5.1 Challenges

We had to create a decay principle because there were patches of unexplored area covered by explored area regions on all 4 directions but this caused it to visit the same area more than two times.We had to test the simulation for different decay constants as it depends on velocity limit,grid cell size.(Obstacle cell is not decayed)

Initially we determined velocity by inverse squared law and this caused an extreme dispersion among the robots to the edges of the Map hence we have set a minimum distance after which the repulsion force between the agent and its neighbouring agent will be removed.

When we were creating a central node that runs all the decision processes we observed that there was a time difference between the publishing of velocity (Vx,Vy)[This caused an oscillation in the robots as it was overshooting the threshold region of the exit cell.] and the moving of the robot this was resolved by creating four seperate ROS nodes that subscribe and publish to a central topic.

Initialy we took a grid cell size of 0.5*0.5 which caused our bot to overshoot its threshold this was resolved by taking a 1*1 cell.Moreover when we take grid cells of more size it consumes less memory,and it takes more time for decision making but reduces cleaning efficiency so it is a tradeoff.

5.2 Simulation and Debugging

We have used Gazebo 9 in the ROS-Melodic in Ubuntu 18.04 in different PC's.We created four ROS nodes which publishes and subscribes from a common topic known as /map.The velocities are then calculated based on the decision making process and thereby published to the /cmdvel topic of the individual robots.The output was viewed in rqt image viewer as 100*100 array out of 50*50 cells were the map and remaining are just kept to account for obstacles.

We have used a planar move,and lidar plugin to gain information about the obstacles in the environment when implemented in the real world mecanuum wheels in specific wheel velocities can move without changing orientation in a plane.

We faced issues as we couldn't simulate 4-robot computing behaviour in the PC we were using and hence there was a lag in the publishing of velocity and often the simulation used to freeze.