# Agent Based System Design

Khandker, Syed Ibrahim

syibkhan@students.jyu.fi

University of Jyvaskyla, 40100, Finland.

## 1. Introduction:

Our aim to design a robot vacuum cleaner to clean domestic environments daily. The robot need to be intelligent enough to operate itself to move, detect, suck the dust, avoid obstacle. In case of any unexpected situation it should have the capability to reconfigure itself finally if not possible inform the owner about the state. Robot will be operating on horizontal surface, with normal earth gravitational force. Position of obstacle, length x width of the environment, temperature, battery lifetime, amount of dust, friction of the surface would may have effects on the operation. The main target of the robot is to be autonomous, that's mean it should have the ability to create its own rules of behaviour. It may seems very easy for human to do a simple cleaning job but if that human is blind than complexity increases a lot. Our robot doesn't have neither eyes nor brain, we may be able to give eye sight to it by sensor but still brain is missing; "Artificial Intelligence" would be using to make decision. The robot should accomplish it job without any pre defined mark on the floor for positioning purpose or beacon at operating area as it has to work in different unknown environments.

## 2. Hardware Design

The design part of the robot can be divided into two major part, Logic and hardware.

**List of Hardware**

- Structure
- Sensors
- Actuators
- Compass
- Battery

Since we don't know in which shape of the environment it is going to work than it is better to get the information of the environment otherwise robot would waste its time and energy to complete a mission, in an unknown environment if our robot can clean 90% + area that would be considered as satisfy level. To work in such a environment we need perfect combination of our hardware, software and sensors, for example reaction of sensing chair leg and wall cannot be same, robot needs to know it's hardware dimension whether it can go under the chair or not. This integrated design based robot is often called "Application Specific Mobile Robot" (ASMR).

## 2.1 Structure of the Robot

To steer the robot intelligently (by little space and use of less energy) specially when it is in some wall corners or unknown type shapes obstacle "circular" shape is very effective. Rectangular shape robot need to come back and change the driving direction. Almost all commercial vacuum robot vendor e.g. LG, Samsung produce circular shape robot.

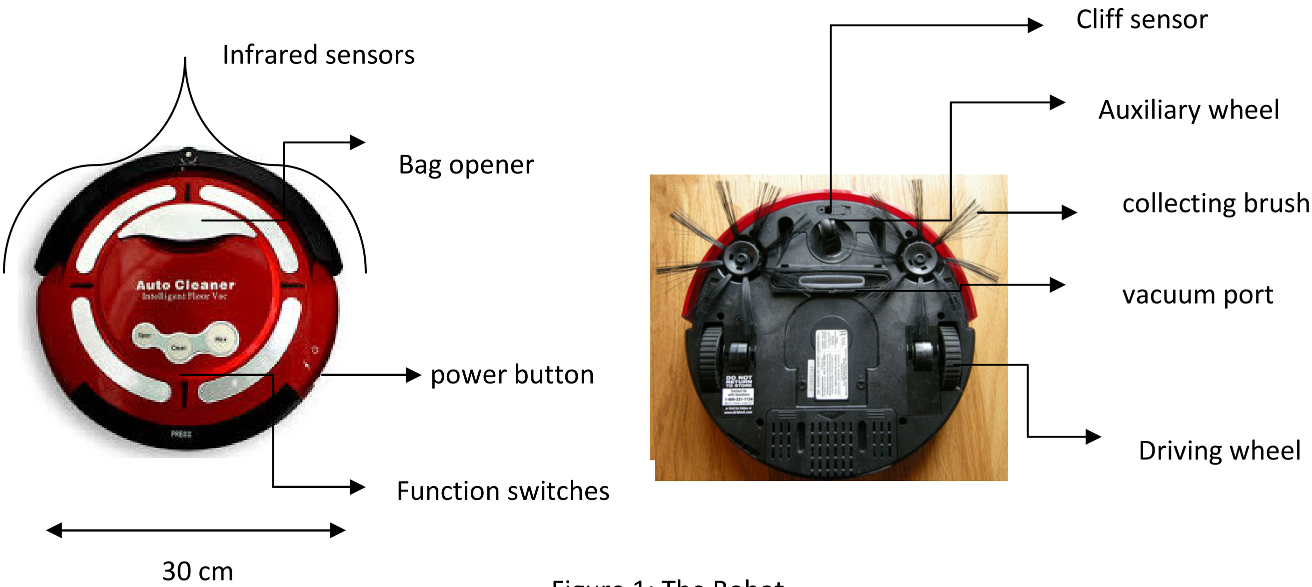



Figure 1: The Robot

## 2.2 Sensors

The purpose of sensor with regarding name below:

**Infrared sensor** is used to determine the obstacle and distance between robot and obstacle, in this case Infrared detectors will also be used. Problem of infrared sensor is it cannot sense transparent obstacle e.g. Glass of water. Ultra-sound sensor can solve the problem but for the sake of simplicity infrared sensor has been chosen.

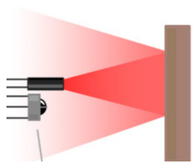


Figure2: Infrared sensor

**Cliff Sensor** would be used at the bottom of the robot to protect robot from falling down while it is in the edge of the floor.

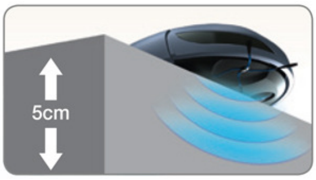



Figure3: Cliff sensor

## 2.3 Combination of Hardware

Two differential drive wheel will be used for driving purpose while  auxiliary wheel would help the robot to keep balance. Vacuum port  is situated at the bottom where two brush will be feeding the dust in the nearby area of port so that robot can suck the dust. For odometry purpose wheel encoder will be used along with wheel.  Failure of obstacle avoidance algorithm may lead the robot to thrust the obstacle so hard that it may get physical damage, so get rid of the soft bumper will be used around the body of the robot to allow certain amount of thrust before it switch to other behaviour. Infrared  sensors and detector will stay at the roundish boarder of the main structure. A removable dust chamber would stay under the main cover, at the middle part of the robot. If robot fall in such a condition that it cannot handle the situation itself (for example baby or dog set on it) it can draw the attention of its master by making sound through piezo speaker .

# 3. Navigation

Now our robot has necessary elements for moving. It should move according the given instruction (details in Algorithm section).  For a safe navigation some steps need to be taken, those are briefly discussed below.

## 3.1 Controller's objective

- **Stability**, the most important aspect of mobile robot.  Unstable control would be full of wrong data which would led to fail mission, e.g. may be crush into obstacle or room wall.
- **Tracking** capability. How the robot can move around the room.
- **Robustness**. It is the ability of the robot to resist change without adapting its initial stable configuration, we should not be over confident that our system would run all time according given parameters.  One of important aspect of robustness is disturbance Ignorance .
- **Optimality**, how can we achieve our goal in best possible way, for example moving point x to y quickly.

## 3.2 Controller Design:

All movement of the robot need to be controlled for safety and efficient navigation, therefore we need to design a controller what would generate the signal as the robot may need. For example, some time we need constant velocity, some time, Left / right turn, jerky less movement etc. we can define current velocity as x. The robot would change of its position by the time, therefore we can define dynamic D as the value of change of state as a function of time. Reference r can be used to compare changes. We call output y to measure what system has done so far. For any mechanical devices there would be some energy loss, therefore to retain expected velocity we need to know the loss and add that loss with control signal (input u). The total lagging of power to obtain our targeted velocity can be measured by feedback f mapping ( difference of output and reference). Now if we add that feedback with input signal we can retain at our **targeted velocity**.

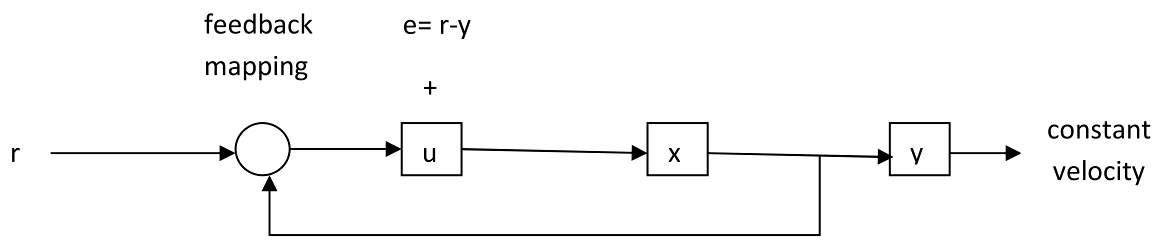



Figure 4: Controller concept

weight of our robot will be changing always ( weight of dust will be added everyday or when we change the dust bag ) so mass m has effect here.

We know from Newtonian law F= ma ............ (equation1)

We also know in any electric car F = cu .............(equation 2)

c = electromechanical transmission co-efficient.

u= input / break

So,

ma = cu

a = c x u / m

dv/dt = c x u / m ................................( equation 3)

This equation would give us effect mass and input signal, if **m** increase more input strength **u** need to be added because **c** is constant. Now this **u** is the source of jerky in robot because error e = r-y always adding here so **e** should be added very smoothly with **u** to **avoid jerky** . By changing reference **r** we can **start / stop** the car. If any dog, baby or other weight is put on the robot **m** shall increase, proportionally **u** shall increase, here we can set a safety value of **u** to halt the robot while excessive mass is applied on it. This is the example of proportional controlling, we know we may have error in accumulation of past errors or prediction of future errors. So we can use very popular PID regulator in our system, so that all type of error would be handled and we shall get a smooth errorless output

$$\mathrm{u}(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t)$$

$K_p$: Proportional gain, a tuning parameter

$K_i$: Integral gain, a tuning parameter

$K_d$: Derivative gain, a tuning parameter

$e$: Error = Refference - output.

$t$: Time or instantaneous time (the present)

$\tau$: Variable of integration; takes on values from time 0 to the present $t$.

## 3.3 Odometry

Since we are using wheel encoder we need to measure how many ticks (N) per revolution. For any certain time period Δtick = New total tick - Old total tick.

Assume

$D_L$ = Distance covered by left wheel

$D_R$ = Distance covered by right wheel

$D_C$ = Distance covered by centre of the robot

$\Delta_L$tick = for left wheel,

$\Delta_R$tick = for left wheel

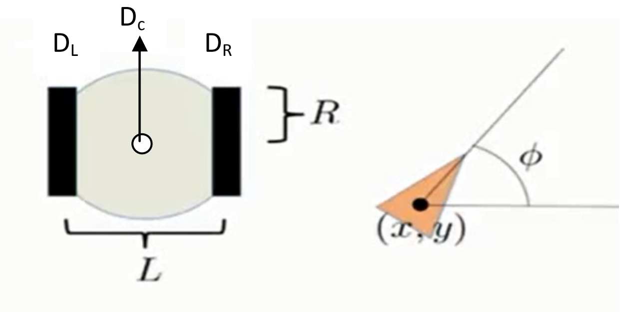


Figure 5: Odometry Concept

Here,

$D_R = 2\pi R\ \Delta_R tick\ / N$

$D_L = 2\pi R\ \Delta_L tick\ / N$

$D_C = (D_L + D_R) / 2$

So,

if old Position of my robot was X,Y, Φ. new position

$X' = X + D_C \cos \Phi$

$Y' = Y + D_C \sin \Phi$

$\Phi' = \Phi + (D_R - D_L)/L$

## 3.4 Obstacle Identification

Sensor can sense the obstacles, there will be many kind of obstacle, robot should behave differently for each type of obstacle to covering whole environment. Since the robot will be working in unknown environment, classification of obstacle may help robot to create a map and set behaviour for that obstacle in future if robot is assigned for same duty, beside this different obstacle can be used as landmark at the navigation process, for example we cannot set up small furniture as a landmark as it is movable but wall can be the landmark; the robot will be able to determine its position by recognising that obstacle what was familiarized during the exploration of the unknown environment. I decided to divide obstacle into following categories

**Furniture Leg** - Obstacle with diameter less than 20cm ( may be table, chair, sofa or other furniture's leg)

**Wall** - obstacles with straight edge. Big furniture e.g. cupboard, wardrobe go under this group.

**Corner**- Meeting point of two separate obstacle.

**Unidentified**- All the obstacles that do not fall in any mentioned group.

## 3.5 Hypothesis

First time any environment will be unknown for the robot, but if we save the map of the room later on it would be easier for the robot to make decision easily, based on obstacle information our robot can make some hypotheses to make a coherent map. Hypothesis shall be verified to eliminate error due to incorrect identification. For example when the robot identifies an obstacle as a wall it can make the hypothesis that the wall is straight by travelling alongside the wall robot can measures the length of the wall by odometry. It also may happen that the hypothesis is not correct, for example if any wall straight at the beginning and ending but curve at the middle, in this situation the robot will identify it as a wall in a first step, but after having followed the wall, the robot will detect that its odometry changed too much for an expected straight trajectory and it will reject the hypothesis and marked that obstacle as unidentified object. Concerning angles robot may need to make hypothesis, because normally the angle between wall in human apartment is 90 degree, and this angles make very good sense to draw a coherent map therefore for angles between 85 to 95 degree the robot will assume that the angle is 90 degree. If the real value of the angle ±5 degree than 90 degree we can still assume that as 90 degree because the wall inside apartment is not so big that that would lead us to a big error. If wall is very long we can shift to exact value.

## 4. Mapping

A correct map ensure the effective cleaning, map would provide information to robot that which areas it need to go through. The most challenging is to keep the map coherent, it must represent the obstacles relatively in correct position. our job would be easier if we make adumbration of the room first than explore it's interior.  This contour map can be used as the reference of the obstacle which will be identified inside the environment. With help of this outlined boarder map robot can get the info of its position so common mobile robotic problem  "**Getting lost into the Environment**" would be solved also. Wherever it face obstacle it would change its direction, the coverage area would be saved in memory, so robot know that mapping of that area has been completed, by this way once robot would find that accessible all area in its memory what would indicate competition of mapping. At mapping phase if it travel many times through same area that would not be the a big problem ( only waste of energy). As it is machine process due to sensor's error we may produce some error in map what need to be corrected later on.  For example, according our hypothesis the meeting point of two perpendicular wall should be same, based on this rule we can eliminate some error, however there could be some error which has no effect on our operation or due to disturbance there can be some little error what is not possible to eliminate.

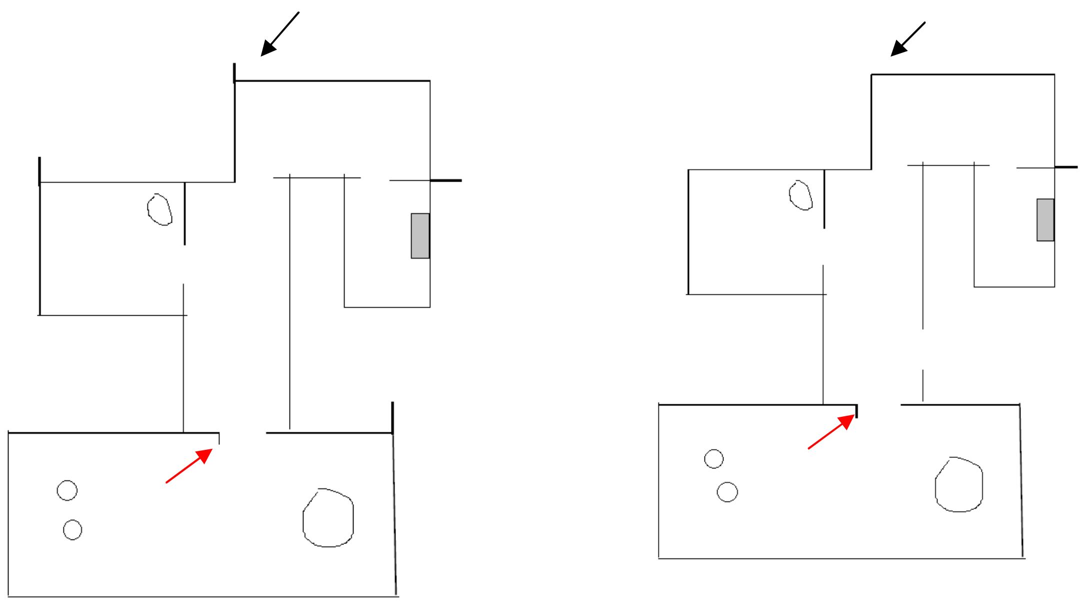


Figure 6    a) Before correction                                        b) After correction

In the figure 6 the black arrow shows that the correction of error, the red arrow shows due to some disturbance little error may happen what is not possible to  correct due to lagging of instruction in our algorithm.  Some trial and error method during real experiment can help us to get rid of these problems.

We divided obstacles into 4 groups, out of that the "Corners" can be potential helpful item for us since it cannot change place (by putting obstacle in front of the corner the position can be make inaccessible but hopefully all corners will not be like that). In a list we can save corner id and X,Y, Φ position. Due to drift wheel encoder may produce wrong data, at operational period by the reference of the corner our robot can recalibrate it's odometry to know it's correct position.

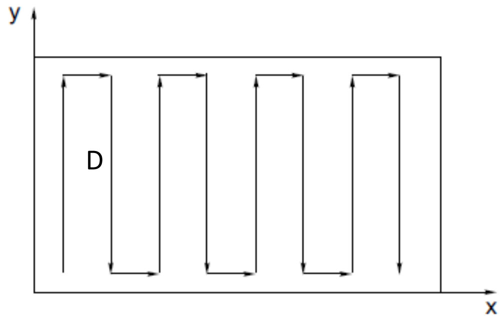
## 5. Instruction

1. If cliff sensor sense any stair don't go ahead, mark that point as not accessible.
2. Distance between two obstacle < Robot diameter + safety distance (1cm) mark that point as not accessible.
3. During mapping position of corners will be saved in a list for that corresponding environment.
4. For any operation (mapping / cleaning), the nearest corner point will be saved as the origin point.
5. After mapping, that area will be saved as current environment, if user want to put robot in a new environment he/ she should select the "new environment" option than robot would start mapping first than cleaning. If user forget to select new environment option that would lead the robot to get wrong data regarding obstacle. If more than 50% corners are explored with new position robot itself assume that environment as a new environment so it will start mapping.

## 6. Algorithm

Now robot know it's job and environment, now we need to develop path planning technique so that robot can cover the surface with minimum travel. It is very important because if we cannot do it robot may travel in the environment until it's battery get finish; path planning is also very much challenging for our task as the environment is unstructured and obstacle may change position. With help of " Mobile Robot Navigation with Complete Coverage of Unstructured Environments " by E. Garcia, P. Gonzalez de Santos [1] we shall try to implement the path-planning algorithm for unstructured environment by detecting critical point. The robot shall start its job from **point of origin** ( any nearest corner, Critical Point 1), start back and forth motion by detecting wall

D = Diameter of Robot




If the sensor detects an object in the robot's path, then the robot changes its trajectory to follow the object's contour until it cover the diameter of the obstacle and create another critical point (CP2). At this point Cell (C1) is cleaned and two more cell (C2 and C3) and IN point is created. If the robot travels a horizontal distance around the obstacle at the very last point of the obstacle will be marked as CP3 and OUT point will be created. Robot goes around the obstacle back to the location where the obstacle was detected and continues sweeping the cell with back-and-forth motions. While it reach the OUT point it closes two existing cells, C2 and C3, and opens one new cell, C4. When the last corner in the field (named CP4) is found, then the robot will be guided to any critical point with disconnected diverging edges because that represent that cell is not cleaned (visited). Adjacency graph d shows us that ( left - right sequence )

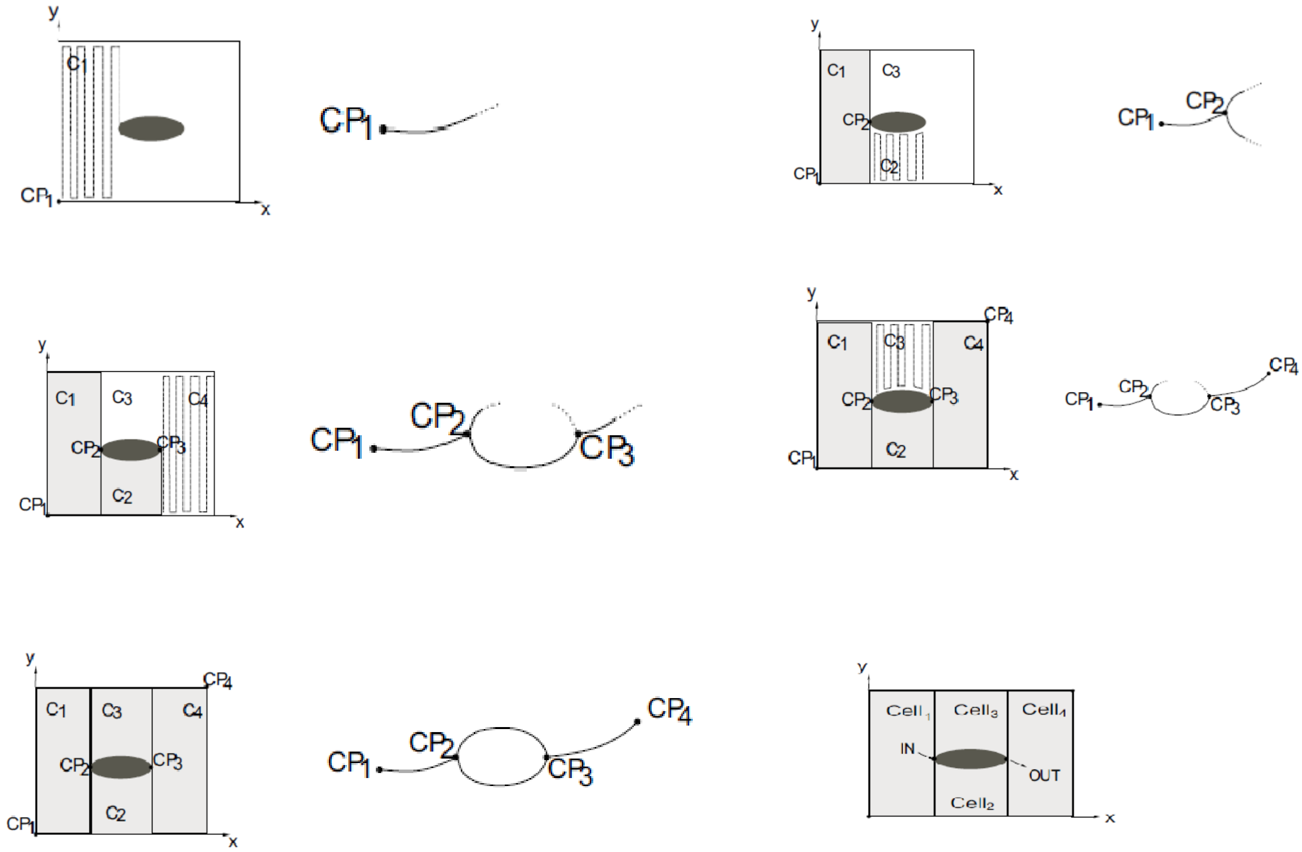



Figure 7: Cellular decomposition and adjacency-graph construction

This is the basic concept of covering areas with obstacles, but there will multiple obstacle in the surface so we can extend our concept for multiple obstacle

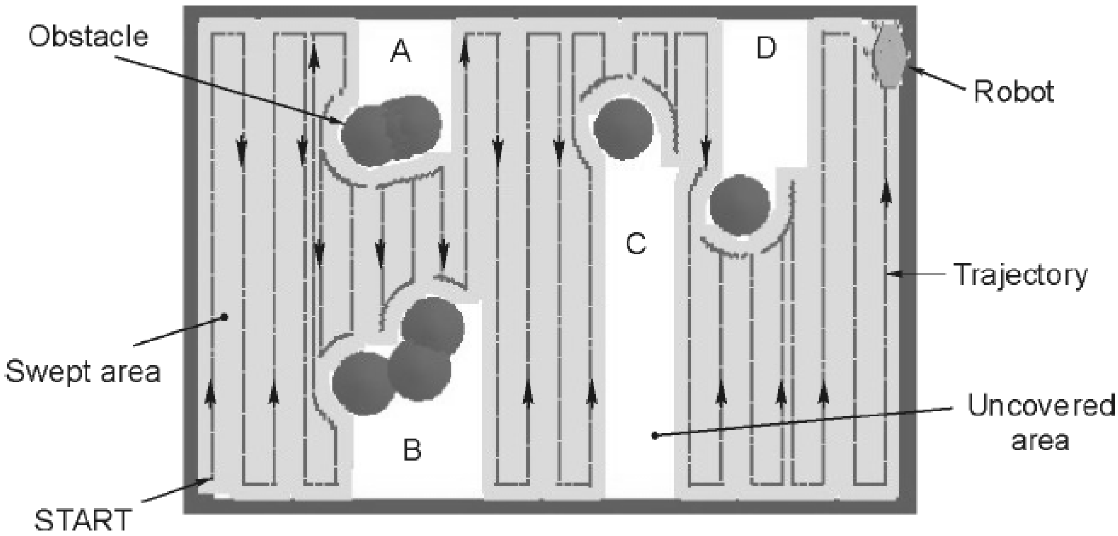



Figure 8: Problem to clean due to obstacles

In the figure 8 we can see the robot has tried to cover a surface with multiple obstacle but due to the scatter position of obstacles some pocket has been created what need to be clean. Now the question is how can robot cover these unclean pocket areas with minimal drive?

We can set the set of unclean Cell as S. For each obstacle we have IN, OUT point; X, Y, position of IN , OUT point we can get from odometry. If robots current position is ( $X_R$ ,$Y_R$ )

**DO WHILE** number of unvisited IN > 0 in the adjacency graph

Step 1: Compute the shortest path to the last unvisited IN point on the adjacency graph.

Step 2: Follow the shortest path to the IN critical point.

Step 3: Sweep the current polygonal area ($S_N$) until the corresponding OUT point is found.

Step 4: Remove ($S_N$) from the set S of disjoint polygonal areas. Remove the last IN critical point from the adjacency graph.

Step 5: SET $X_R = X_{OUT}$ , $Y_R = Y_{OUT}$ . Go to Step 1.

if unvisited IN=0; Go to **point of origin** (X,Y, Φ)

## 7. Conclusion

It is really a tough task to design a system with only theoretical information, I just tried to explain how automatic vacuum robot can be built. For implementing this in real life definitely we shall face many difficulties because as a human being we always thing like a human being but thinking capability of machine is zero unless we put some logic there. This assignment has helped me to think like a machine.

## Reference:

1. E. Garcia, P. Gonzalez de Santos. Mobile Robot Navigation with Complete Coverage of Unstructured Environments. http://digital.csic.es/bitstream/10261/12814/3/elena-garcia_coverage_final-1.pdf

2. Online course " Control of Mobile Robots" . Georgia institute of Technology. https://www.coursera.org/course/conrob