A Cost Effective Probabilistic Approach To Localization And Mapping

Dibyendu Mukherjee†, Ashirbani Saha†, Pankajkumar Mendapara†, Dan Wu‡ and Q.M. Jonathan Wu†
†*Department of Electrical and Computer Engineering*
‡*School of Computer Science*
University of Windsor, Ontario, N9B 3P4, Canada
{*mukherjd, sahai, mendapa, danwu, jwu*}*@uwindsor.ca*

Abstract

Localization and mapping in robotics are preliminary but challenging problems. A learning approach must be followed by a robot to understand its environment and perform data association before it accomplishes any other tasks. In this paper, we describe a novel combination of techniques to map the environmental boundaries traced by the robot and localize it inside the bounded region. This is an effort established using only an iRobot educational package and no expensive high-end external sensor. This method may be treated as a solution for mapping and localization in a static environment with a few low cost IR sensors. In the proposed approach, we trace the robot's movement in an arbitrary shaped bounded region and map the same using coastal rule wall following technique and the method of least squares. A full traversal of robot maps the boundary and the robot is localized in the environment using particle filter approach and computational geometry. Also, we studied the effect of localizing a kidnapped robot once the map is known.

I. Introduction

Robotics is an interesting and growing research subject and a vast field for advancement and application of artificial intelligence. Using state-of-the-arts techniques and complex hardware, robotics have enhanced beyond the normal belief. Still, research continues to make a cost effective, understandable and learning robot that can be autonomous and perform goals without human intervention. Research involves tracking a robot's motion, localizing it in the region and complete its target. But with increasing level of complexity, the hardware used is more expensive and more precision demanding. Also, the projects are so huge and time consuming that a small scale replica is pretty difficult to design and implement.

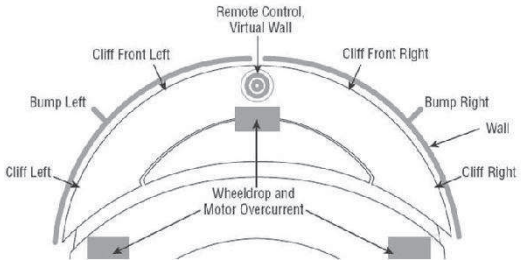


Fig. 1: The iRobot and its Sensors

This paper is about tracing a robot's movement in a bounded region using Coastal rule to follow the boundary walls. The tracing process involves modeling the motion and sensor information of the robot. Simultaneously, an outline map of the environment is created. This map construction process involves boundary sensing, corner detection and noise reduction of sensor data using the method of least squares. The mapping process is complete when the robot has fully traversed the periphery of the entire region once. Finally, the robot is localized in the constructed map using Particle filter based on probabilistic robotics[13]. Probabilistic Robotics, a comparatively new but very popular branch of Artificial Intelligence, represents the uncertainty in robot perception explicitly using the calculus of probability theory. Since the localization process depends on the robot's sensor data which are often noisy and generate uncertainty in estimation of robot position, probabilistic approaches have been successful. The proposition given in this paper is both educational and comparable to real-scale. The entire work mentioned in this paper has been established without any sensor other than the devices provided in an educational robotics package.
There are many commercial robots available for education and research like Lego Mindstorms NXT, Sony AIBO dog,

978-1-4244-3355-1/09/$25.00©2009 IEEE

iRobot Create etc. We have chosen iRobot Create because it is easily programmable and it is available in a pretty low cost package.
The paper has been organized as follows. In Section II, the related works in the existing literature have been discussed. An overview of the iRobot and the sensors used are provided in the Section III. In Section IV, the application of coastal rule for mapping has been discussed. Section V describes the localization of the robot using particle filter and computational geometry. Section VI provides the experimental setup and results. The concluding part is Section VII which discussed the scope for future work.

II. Related Works

Robot localization has been considered as - "the most fundamental problem to providing a mobile robot with autonomous capabilities" [5]. The simplest localization problem is the position tracking, where the initial pose, the location and orientation of the robot is known, compared with the global localization problem, where the orientation and the location of the robot is not known. One of the popular algorithms used for position tracking is "Kalman Filter" [10]. The "Particle Filter" or "Monte Carlo Localization" as it is known in robotics, is one of the simplest yet efficient algorithm that solves the global localization problem and also the kidnapped robot problem where the robot is shifted to an arbitrary position while it considers itself to be properly localized. Recent works in [11], [8] show these methods used in mobile robotics to solve both the position tracking and global localization problems. Also, besides the localization problem, mapping the environment explored by a mobile robot is another well known fundamental problem in robotics. If the robot's locations are known, the mapping problem can be solved as described in [3]. The solution for iterative mapping and localization involved the use of expensive sensors (high range sonar or laser sensors) as shown in [6], [12].
Our paper proposes a novel combination of the computational geometry and particle filter for robot localization in a low cost platform and also focuses on coastal rule and the method of least squares for mapping. There have been many approaches in cost reduction for mobile robotics. The papers [14] and [16] discuss the fabrication and usage of a cost effective educational robot. The effectiveness of "Roomba" (an autonomous vacuum cleaning robot) as a research robot was shown in [2] where Roomba was concluded as "a promising alternative to many other low cost robot platforms available for research and education". The methodology of accessing Roomba's sensor data was made easy by [15]. iRobot Create, the robot used for this paper, is a special adaptation of Roomba and designed mainly for research and educational communities.

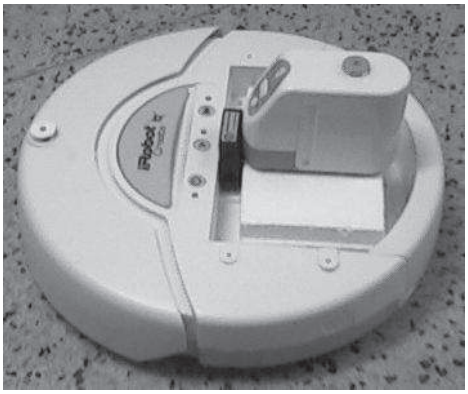


Fig. 2: The iRobot Create with the Virtual Wall mounted on it

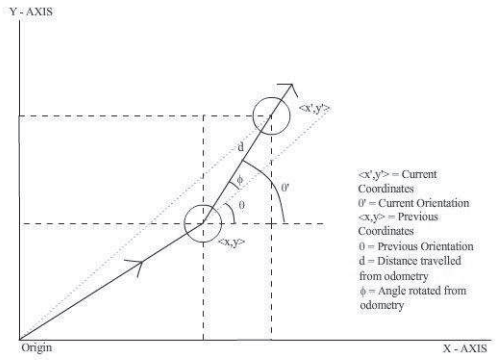


Fig. 3: The Motion Model of robot

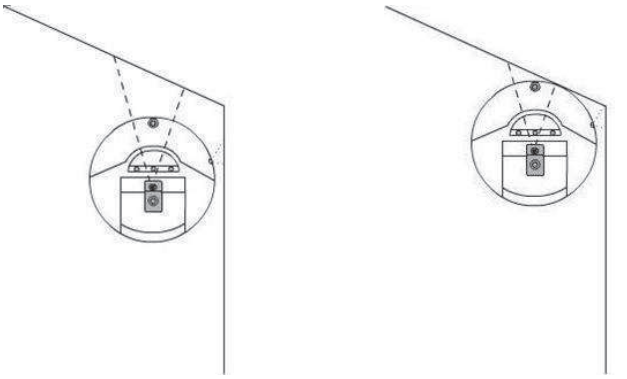
III. Overview of the iRobot

The iRobot is shown in the Fig 2. Most of the sensors of the iRobot are located at the front end and the schematic view of the same is presented in Fig 1. The bump sensors are used to detect any collision at the front end. The cliff sensors are used for detecting any uneven surface or cliff. The wall sensor can sense the presence of a wall at the right side and this is the reason the robot can be used to follow wall keeping it on its right side. In addition to these sensors, the package comes with two virtual walls, a command module (flash memory size 14336 bytes only), advanced battery power system, standard remote and self charging home base with fast charger. The virtual wall is generally used to restrict the movement of robot in a specific direction but for our method, one of the virtual walls has been mounted on iRobot (shown in the Fig 2) for corner detection. The virtual wall is an IR emitter and iRobot can detect its presence using the IR detector at the 12'0 clock position (shown in Fig. 1).

IV. Tracking Movement and Map Generation

In this section, we discuss how we track the robot's movement and map its environment.
The robot starts from an arbitrary position within the region. Then, it seeks the closest boundary point with respect to its heading direction. After it finds the first boundary point, the robot follows the boundary keeping it on its right. In each step, the current location of the robot is added to the list of locations it had been. This incrementally sketches the map of the region. After a full traversal, the robot stops following the wall. As the region is bounded, the starting and ending positions are almost same if the robot completes a full traversal. This generates a complete map of the bounded region.
For a mobile robot, the current kinematic state or the pose is very important. For a planar environment like in ours, the pose or state is described by the current Cartesian coordinates (x, y) of the robot along with its orientation. Orientation is the heading direction of the robot and normally it is denoted as the angle (θ). The state $(x, y, \theta)^t$ is obtained in each time step using the kinematics model of the robot motion. This model is referred to as the Motion Model. Two types of Motion Model are prevalent in existing literature: the Velocity Motion Model and the Odometry Motion Model. In the Velocity Motion Model, the velocity information is obtained from the robot sensors. In the Odometry Motion Model, the odometry information (the distance travelled and the angle rotated by robot) are obtained from sensor. We have chosen Odometry Motion Model for iRobot.
Similarly, we model the sensor data of the robot and this is called as Sensor Model. The sensors are already discussed in Section III. A sensor model is used to model any uncertainty introduced by the imperfection of sensors while estimating state from the motion model. The uncertainty occurs due to measurement errors in the odometry sensors and the non-uniform or uncertain nature of environment. The modelling of uncertainty is the core of motion model in Probabilistic Robotics. Due to the deterministic coastal rule approach used by robot, the uncertainty decreases by a large amount and so the motion model is simple for the current section. When the region is not known to the robot and it follows the wall, the motion model does not use any probabilistic approach. It simply estimates current state of the robot by correcting the odometry information from robot's sensors, using least square method. We refer to this motion model as "Motion Model for Mapping" and similarly, the sensor model is termed as "Sensor Model for Mapping".

In this paper, the state estimates obtained from the Odometry Motion Model are combined to construct the outline map of the environment. So, it becomes necessary to make the robot move along the boundary of the region. This is accomplished by using the Coastal Rule technique discussed in Sub-section IV-C.



(a) Virtual Wall senses wall

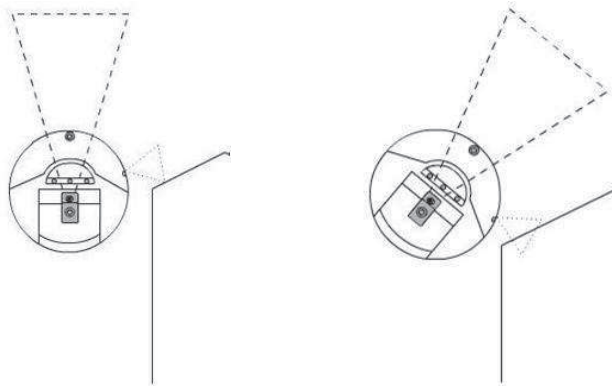
(b) Bump Sensor detects a wall

Fig. 4: The Virtual Wall senses a front wall in (a). But the distance is not obtained from this reading. In (b), the bump sensors get activated when the robot collides with the wall. Hence, a non-reflex corner is detected.

The boundary of the region can be arbitrary shaped. In real life, the best estimate for an arbitrary shape is a polygon. Even, the circle can be thought of as a polygon of a large number of sides. As two consecutive state estimates construct a side of the map, more number of relative state estimates (of the robot) close to each other, will yield a better but computationally expensive map.
While tracing out the boundary, the corners are detected and stored. Corners are very important for the simplified processing of the map. Further justification for the usage of corners will be discussed in Sub-section V-C.
Also, it is very important to make the robot understand that it has completed a full traversal of the boundary. The first position where the robot detects a wall is at some distance from its origin on the x-axis. At any position (x, y) close to boundary, the $\arctan(y/x)$ represents the angle covered by the robot w.r.t the positive x-axis. When this angle equals or exceeds 360°, it ensures completion of a full traversal.
A brief description of the models and techniques used to establish "Tracking Movement and Map Generation" is discussed as follows.

A. Motion Model for Mapping

The motion model is based on the angle and distance from robot's odometry. The initial position of robot is assumed as (0, 0) and orientation as 0°. It is important to note that current angle and distance are relative to the robot's previous position. Using this data the current state of robot can be estimated. The estimation procedure is



(a) Wall Sensor can not detect a wall

(b) Robot turns right to find wall

Fig. 5: The wall has turned right and wall sensor loses a reading as shown in (a). The robot turns right to find the wall and if the turn exceeds a threshold, a reflex corner is detected as shown in (b).

explained in Fig 3.

The robot's odometry sensors provide values of the distance traveled (d) and the angle rotated (ϕ). At any point in time, the robot's co-ordinates are (x, y) and the orientation is θ. The robot updates its distance and angle after each movement but the values are needed to be collected after an optimum interval. If this interval or pause time is large, the values will cross the storage capacity of the registers of robot and will be flushed. Again, for too low pause time, the robot cannot supply the values due to a latency in its processor. So, after an experimentally optimum pause time (70 ms), the values of d and ϕ are obtained, and the resultant co-ordinates and orientation of the robot are now

$$x' = x + d * \cos(\phi + \theta) \quad (1)$$

$$y' = y + d * \sin(\phi + \theta) \quad (2)$$

$$\theta' = \phi + \theta \quad (3)$$

During the application of coastal rule, due to imperfection in sensors and uneven surfaces, the odometry data from the robot sensors deviate from the actual values. The angle turned (ϕ) by the robot depends on the direction of wall but the distance travelled (d) is totally unguided. So, d is significantly affected by noise compared to ϕ and needs correction. Before using d, it is refined by method of least squares because the actual distance travelled ($\tilde{d}$) is different from the distance obtained from robot odometry (d). A set of data of ($\tilde{d}$) versus (d) is obtained a priori by experiment. Here, $\tilde{d}$ can be written as f(d) and a straight line is fitted to this data such that

$\tilde{d} = a * d + b$

a and b are calculated as follows -

$$a = \frac{\left(\sum \tilde{d}\right)\left(\sum d^2\right) - \left(\sum d\right)\left(\sum d\tilde{d}\right)}{n\sum d^2 - \left(\sum d\right)^2} \quad (4)$$

$$b = \frac{n\sum d\tilde{d} - \left(\sum d\right)\left(\sum \tilde{d}\right)}{n\sum d^2 - \left(\sum d\right)^2} \quad (5)$$

Hence the changed equations during the application of coastal rule are

$$x' = x + \tilde{d} * \cos(\phi + \theta) \quad (6)$$

$$y' = y + \tilde{d} * \sin(\phi + \theta) \quad (7)$$

$$\theta' = \phi + \theta \quad (8)$$

Here, $\tilde{d}$=refined d although ϕ remains same.

B. Sensor Model for Mapping

The sensors used for the process of mapping are the wall, virtual wall and the bump sensors. The wall sensor and the virtual wall sensor are used to assist the proof of existence of a wall or a corner. The robot will face a corner in a bounded region even if the region is a circle. In case of a circle, the region will be approximated by the robot as a polygon with few corners. The corners are non-reflex ($<180°$) and reflex ($>180°$) in nature.

When the virtual wall and bump sensors simultaneously get activated together, the event is registered as "Non-reflex corner detection" as shown in Fig 4.

When the wall turns right, there is a reflex corner. At this point, the wall sensor gives a 0 reading and the robot turns right to find the wall again. If the turning angle exceeds a threshold, the event is registered as "Reflex corner detection" as shown in Fig 5.

C. Coastal Rule

As pointed out in [8], while navigating sufficiently close to coastlines with high information contents, the likelihood of getting lost can be minimized for ships in absence of better tools like GPS. The same fact has been verified using a museum robot Minerva and discussed in [8]. In case of the iRobot, it does not have laser, sonar, GPS or similar type of sensors. But, it has a built-in wall sensor for boundary sensing purposes. Using this simple sensor along with the bump sensors, the Coastal Rule had been implemented successfully. The Algorithms used are:

Algorithm Coastal_rule(x_0)

1 : $Find\_first\_wall(x_0)$

2 : $map = Follow\_wall(x_t)$;

3 : $return\ map$

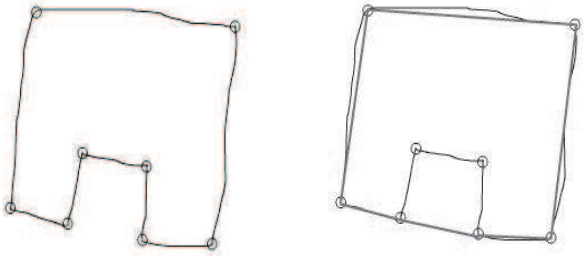
Algorithm Find_first_wall(x_{t-1});

```
1 :  speed = 60;
2 :  foundwall = false;
3 :  while(NOT foundWall)
4 :      if(bumpLeft() OR bumpRight() OR wall)
5 :          foundWall = true;
6 :          break;
7 :      else
8 :          u_t = goForwardAt(speed);
9 :          x_t = motion_model_odometry(u_t, x_{t-1});
10 :     endif
11 : endwhile
12 : return (x_t)
```

Algorithm Follow_wall(x_{t-1});

```
1 :  deg = 25;
2 :  speed = 60;
3 :  map = ∅;
4 :  turnSpeed = 200;
5 :  while(NOT ((user_stop) OR (arctan(x_t[2]/x_t[1]) ≥ 2 * π))
6 :      if(bumpLeft() OR bumpRight())
7 :          u_t = spinLeft(deg);
8 :          x_t = motion_model_odometry(u_t, x_{t-1});
9 :      elseif(NOT wall())
10 :         u_t = turnRightAt(turnSpeed);
11 :         x_t = motion_model_odometry(u_t, x_{t-1});
12 :     else
13 :         u_t = goForward(speed);
14 :         x_t = motion_model_odometry(u_t, x_{t-1});
15 :     endif
16 :     map = map + x_t;
17 : endwhile
18 : return map
```

The Coastal_Rule algorithm takes the origin of robot (0,0) as input and calls two functions - Find_initial_wall() and Follow_wall(). Find_initial_wall() makes the robot go forward till it finds a wall and returns the coordinates of the wall. Here, x_t represents the pose $(x, y, \phi)^t$. The motion_model_odometry() function calls the motion model of Sub-section IV-A with the action u_t and previous coordinates. The output are the current coordinates. The Follow_wall() algorithm takes the first wall coordinates and continues drawing the map by adding each coordinate of the robot to the map. There can be three cases as shown: the robot can find a wall in front (a non-reflex corner), no wall (a reflex corner) or it will go forward following wall. After completion of 360 degrees traversal or if a user breaks the loop, the map is returned. Using the motion model and coastal rule, an outline map of the region, in which the initial location of robot is known, is generated. Sample maps of its environment drawn by iRobot are shown and discussed in Section VI.



(a) A Sample Map (b) Approximated Map

Fig. 6: (a) shows a Sample Map with reflex and non-reflex corners. (b) shows the approximated simple polygon map by connecting the non-reflex corners

V. Localization of Robot in Constructed Map

This section describes the localization of the robot inside the created map. We use Particle Filter for robot localization. Particle filter, as already mentioned in Section II, is from the bayesian filter family that are used to decrease the uncertainty present in a process using probabilistic algorithms. The Coastal Rule is a very deterministic process, but for a real-life application, a boundary following robot would not suffice. The robot needs to be localized inside the map, not only along the boundary. After generating the map, the robot stops the wall following, and moves in a random manner to explore the space enclosed inside the region. Any zig zag motion would introduce uncertainty in the robot movement and this uncertainty must be modelled in the motion model itself. So, the motion model is modified and will be described in the next sub-section. The sensor model also needs to change because the wall sensor is no longer used and the uncertainty also needs to be incorporated. The Particle Filter is well described in [4] along with a few resampling techniques. We describe in this section, the use of particle filter and computational geometry for localization and give a brief note on the kidnapped robot scenerio.

A. Motion Model

When the robot has the map and starts localizing itself, the uncertainty increases, and the least square approach is not applicable anymore. The reason is that the robot is no longer guided by a wall and both the distance travelled (d) and the angle rotated (θ) are now subject to sensor and environmental noises. The noises result in increase of uncertainty in both the parameters that cannot be corrected by simple least-squares. The uncertainty can be estimated if the probability distribution of current state x_t is known, which is not always possible. So, sampling is used for the purpose. A sample of x_t is estimated from the previous state x_{t-1} and the current action u_t. The problem is to estimate p(x_t) using p(x_{t-1}) and action u_t performed by

the robot . It is same as finding p($x_t \mid u_t, x_{t-1}$) taking into account the uncertainties in robot odometry. We introduce Φ and D in place of ϕ and d respectively to model the uncertainty in robot motion through equation (9) and (10)

$$\Phi = \phi - sample\_normal\_distribution(\alpha 1 * d + \alpha 2 * \phi) \quad (9)$$

$$D = d - sample\_normal\_distribution(\alpha 3 * d + \alpha 4 * \phi) \quad (10)$$

Equations (4), (5), (6) are changed to

$$x' = x + D * \cos(\Phi + \theta) \quad (11)$$

$$y' = y + D * \sin(\Phi + \theta) \quad (12)$$

$$\theta' = \Phi + \theta \quad (13)$$

$\alpha 1$,$\alpha 2$, $\alpha 3$ and $\alpha 4$ are the mean errors in robot's odometry. The robot cannot move a pure rotation or translation. The amount of translational error in rotation ($\alpha 1$), rotational error in rotation ($\alpha 2$), translational error in translation ($\alpha 3$) and rotational error in translation ($\alpha 4$) were determined experimentally. The sample_normal_distribution generates a random sample from a zero-centered distribution and takes input as the variance. The algorithm [13] is as follows-

Algorithm sample_normal_distribution(b);

1. *return* $\frac{b}{6} \sum_{i=1}^{12}$ *rand(-1,1)*

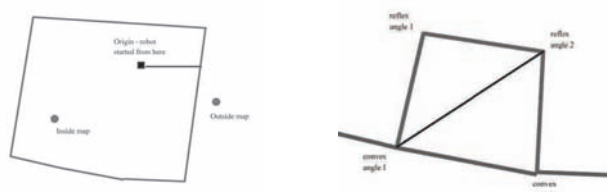
Now, the robot's position can be presented as a set of probable points instead of a single point and this depicts the uncertainty of the robot.

B. Sensor Model

The sensor model now uses only the bump sensors and the virtual wall sensor for detection of wall and corner. Repeated testing of the virtual wall sensor showed that it has a standard error in locating an obstacle. During the application of Coastal rule, the combination of bump sensors and wall sensors alongwith the dependance on the walls helped to minimize the uncertainties. Now, due to the absense of guided motion, the detection error probability of the virtual wall sensor must be modelled. The error has been modelled as a gaussian distribution with zero mean and variance σ = 0.8 (experimentally obtained) using a standard normal variate table.

C. Use of Particle Filter and Computational Geometry

The Particle Filter approximates the state of the robot by a number of particles. It has two parts. The first part approximates the current state of each particle using its previous state and the current action taken by the robot. Then the particles are weighed using the error probability of the sensor discussed in Sub-section V-B. The second part consists of resampling the particles from this approximated state using a rule - the particles are sampled according to their weight. The heavier (hence, more probable) particle are chosen more and the lighter particles are chosen less number of times. As this is a "Select with Replacement" sampling, a particle can be chosen multiple times. A number of resampling techniques are given in [4].



(a) Deciding whether point is inside the convex polygon containing the origin

(b) The convex Polygon constructed using the points responsible for reflex angles in the original map

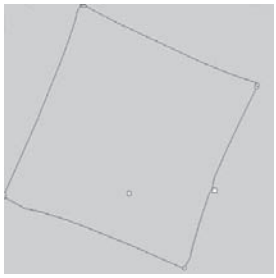
Fig. 7: (a) shows the process of accepting or rejecting a particle, (b) shows the polygon constructed by the reflex corners

The generated map of the region is arbitrary shaped as shown in Fig 6a. The robot's sensor data gives an estimate of its position with respect to the map. The particles close to the position are more probable to be resampled. In any step, the particles going outside the map must be neglected because the robot necessarily stays inside. The entire particle selection process has been successfully implemented using computational geometry.

For this, each particle is compared to the boundary of the map in each time step. The map consists of coordinates of the boundary connected by straight lines. Each coordinate was collected from the position of the robot at each time step. So, there are a huge number of points to handle. To minimize the number of points, the corners of the map can be joined to approximate the original map. But, there may be reflex and non-reflex types of corners that need different treatment. We split the approximated simple map in two parts. In first part, we join only non-reflex corners as shown in 6b. In second part, we treat any consecutive reflex corners as a group and make a convex polygon joining the reflex corners described later. Thus, the corners are very important for the simplification of the map and their detection was already described in Section IV-B.

In the map consisting of non-reflex corners (polygon) obtained, the origin definitely lies within or on it, so any particle lying within or on the map is on the same side of the origin with respect to a particular side of the polygon. For the process, the equations of all sides of the polygon are calculated. Each particle is checked for each equation and if it is on the same side as the origin w.r.t all sides, the particle is kept else is rejected. The process is shown

(a) A region with reflex corners mapped by robot



(b) A square region mapped by robot

Fig. 8: Map traced out by robot for different shapes of bounded regions

in Fig 7a. The particles which overcome the previous step are subjected to the next step which is used only if the map obtained resembles a polygon with reflex angle(s). Here, a mechanism is applied to bring out the points that form the reflex angles. Starting from the initial corner, once such point comes, the point before it and the point after it are joined to it for extracting this triangle out and all the particles (which are inside the triangle) from this set are deleted. For generalization this can be explained as follows: The consecutive (or single) appearances of points responsible for reflex corners are noted. Therefore a convex polygon (without the reflex angles) can be formed using those consecutive (or single) point(s), the point preceding this series and the point just after this series (both represent non-reflex corners). This polygon does not contain reflex angles because all reflex angles are convex for the polygon. The particles which are found to be inside this convex sub polygon are deleted from the set with which the step began. This procedure is followed unless all the corners of the original map are encountered at least once during the traversal from the first to the last point. A number of normal sub polygons can be obtained by following this method. The sub-polygons are triangulated; any point residing in any of the triangles formed is clearly rejected.

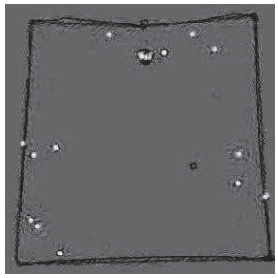
D. A kidnapped Robot

The kidnapping robot scenerio is a special case of global localization as already mentioned in Section II. The twist is, the robot is unaware that it has been kidnapped and still has the belief that it is resting at the last position prior to its kidnapping. The problem is harder than global localization. For the iRobot, whenever it is kidnapped and lifted from the ground, it goes into passive mode and it stays in that mode after it is released. The control must switch the state from passive to active after the robot is put on the ground. So, the control needs to know the exact moment of kidnap and release i.e. a notification of kidnap.

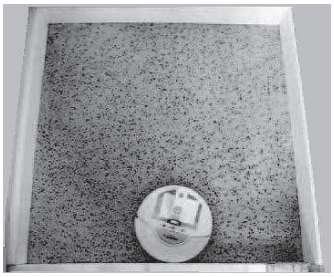
The experimental iRobot has a safety fault mechanism. Whenever the robot is kidnapped, it is being taken off from the ground and the cliff sensors and wheel drop sensors get activated. The activation results in the sudden switch from current mode to passive mode. This also stops the robot motion. Whenever, these sensors are activated, it means robot is kidnapped. The robot is starts again when user puts it back on ground and localization starts.
We could have completely converted the kidnapped robot scenario to simple global localization because robot can be informed by its safety fault that it is being kidnapped, but we tried to replicate the scenerio of localizing a kidnapped robot with our techniques.

VI. Experimental Setup and Results

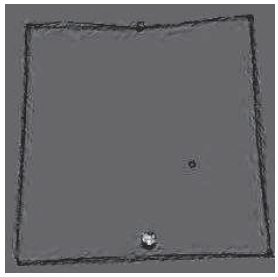
(a) The robot begins localization in the region



(b) The random particles are shown in the map of the region



(c) The localized robot



(d) The particles are concentrated in the most probable position of robot

Fig. 9: The Localization Process

In our experiment, we have used Processing, an open source Java based programming interface, as the platform. In [15], author has discussed the connection and programming of Roomba using Processing and provided precompiled libraries for the same. These libraries are suitable for iRobot also.
For the purpose of boundary and corner detection which are discussed in Section IV, we have mounted the Virtual Wall on top of the iRobot Create. This setup is already shown in Fig 2.The experiments have been carried out in

regions of various shapes and two mappings of the robot are shown in Fig. 8. The dimensions of the maps have been also verified with the corresponding bounded regions.
The $\alpha 1$, $\alpha 2$, $\alpha 3$ and $\alpha 4$ discussed in section V-A, have been found out by averaging the errors obtained over 70 repeated observations. The values obtained are $\alpha 1$=0.3deg/360mm; $\alpha 2$=1deg/360deg; $\alpha 3$=0.026mm/mm; $\alpha 4$=0.0002mm/deg.
The localization of the robot in a map is shown by two picture sets. In Fig. 9a, the robot begins to explore the space enclosed by the boundary. The particle distribution is shown in Fig. 9b. An empty square has been shown on top to designate the starting position of robot inside map. In Fig. 9c, the robot is at the opposite wall and particle distribution in Fig. 9d shows that the particles are also concentrated at that region. At this stage, we can consider that the robot has been localized in the region of maximum concentration of particles. The localization was also confirmed by online visualization and by offline measurement of distance and angle from the origin (the initial position of the robot).

VII. Conclusion and Scope for Future Works

Our paper is a combination of probabilistic techniques with method of least squares, computational geometry and coastal rule addressing some very common yet challenging areas in robotics. The tracking, simultaneous localization and mapping are preliminary tasks of most robot-oriented architectures. A robot has been localized in an unknown environment of arbitrary shape using a few low cost sensors. Even if the shape is very symmetrical (like circular), the robot can still sketch an approximate map of the environment.
The primary goal of localization and mapping is generally achieved by using highly efficient sensors. So, the experiment in a domestic environment with limited resource is difficult. But this has been achieved by our experiments using a single iRobot package without any expensive sensors. This is actually a demonstration showing that using excellent methodologies, small scale reconstruction of an actual real-scale model can be made.
There are a number of scopes of future work. Using range finders for obstacle sensing, the mapping and localization can be improved effectively with our proposed techniques. We are currently working on the same. Also, using better IR wall sensors, an autonomous robot can be created to sketch an unknown territory or maze without any human intervention.

References

[1] A. Björck: "Numerical Methods for Least Squares Problems", *Society for Industrial and Applied Mathematics, U.S.* (1999).

[2] B.Tribelhorn, Z. Dodds: "Evaluating The Roomba: A Low-cost, Ubiquitous Platform For Robotics Research And Education", *IEEE International Conference on Robotics and Automation*, pages: 1393-1399 (2007).

[3] H. P. Moravec and A. Elfes: "High Resolution Maps From Wide Angle Sonar", *IEEE International Conference on Robotics and Automation*, Volume: 2, pages: 116- 121 (1985).

[4] I. M. Rekleitis: "A Particle Filter Tutorial For Mobile Robot Localization", Technical Report, *TR-CIM-04-02*, McGill University.

[5] I.J. Cox: "Blanche - An Experiment In Guidance And Navigation Of An Autonomous Robot Vehicle", *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 2, pages: 193-204 (1991).

[6] J.J. Leonard, H.F. Durrant-Whyte, and I.J. Cox: "Dynamic Map Building For An Autonomous Mobile Robot", *IEEE International Workshop on Intelligent Systems*, pages: 89-96 (1990).

[7] M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf: "Computational Geometry: Algorithms And Applications", *Springer-Verlag, Berlin* (1997).

[8] N.Roy, W.Burgard, D. Fox, S.Thrun: "Coastal Navigation-Mobile Robot Navigation With Uncertainty In Dynamic Environment", *IEEE Internation Conference on Robotics and Automation*, Vol. 1, Pages: 35-40 (1990).

[9] P. Jensfelt and S. Kristensen: "Active Global Localisation For A Mobile Robot Using Multiple Hypothesis Tracking", *Proceedings of the International Joint Conferences on Artificial Intelligence Workshop on Reasoning with Uncertainty in Robot Navigation, Stockholm, Sweden*, pages: 1322 (1999).

[10] R. E. Kalman: "A New Approach To Linear Filtering And Prediction Problems", *Transactions of the ASME-Journal of Basic Engineering*, No. 82 (Series D), pages:3545 (1960).

[11] S. Thrun, D. Fox, W. Burgard, and F. Dellaert: "Robust Monte Carlo Localization For Mobile Robots", *Artificial Intelligence*, Vol. 128, No. 1-2, pages: 99-141 (2001).

[12] S. Thrun, W. Burgard, D. Fox: "A Real-Time Algorithm For Mobile Robot Mapping With Applications To Multi-Robot And 3D Mapping", *IEEE International Conference on Robotics and Automation, San Francisco*, Vol. 1, pages: 321-328 (2000).

[13] S. Thrun, W.Burgard, D.Fox: "Probabilistic Robotics", The MIT Press (2005).

[14] T. Hsiu, S. Richards, A. Bhave, A. Perez-Bergquist, I. Nourbakhsh: "Designing A Low-cost, Expressive Educational Robot", *IEEE International Conference on Intelligent Robots and Systems*, Vol. 3, pages: 2404-2409 (2003).

[15] T. E. Kurt: "Hacking Roomba", Wiley Publishing, Inc (2007).

[16] Z. Dodds and B. Tribelhorn: "Erdos: Cost Effective Peripheral Robotics For AI Education", *in Proceedings of Association for the Advancement of Artificial Intelligence*, pages: 1966-1967 (2006).