Sonar sensor virtualization for object detection and localization

Conference Paper · April 2016

DOI: 10.1109/SECON.2016.7506699

2 authors:

Md Hossain Shuvo
Auburn University
6 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Yujian Fu
Alabama A & M University
40 PUBLICATIONS **73** CITATIONS
SEE PROFILE

Sonar Sensor Virtualization For Object Detection And Localization

Md Hossain Shuvo, Yujian Fu
Department of Electrical Engineering & Computer Science
Alabama A&M University
Normal, AL 35762
shuvo108590@gmail.com, yujian.fu@aamu.edu

***Abstract*—** Localization and obstacle detection have been unremittingly a challenging issue in the field of robotic from the past decades because of whimsical preciseness as well as cumulated drift of sensorial component and stochastic nature of robot's behavior in real time environment. The concentrated objective of this paper is to adopt the localization and object detection capability for the ground robot with the use of sonar array differently from the conventional sensors arrangement to utilize the magnitude of several sonar sensors and to preserve the port and weight constraint of respective robotic platform. The overall localization and object detection experiment is based on the small scale robotics and local spatial information which has been accomplished in a 2D geometrical grid map. For localizing the odometry data directly fetched from the servo motor has been used and for the object detection we have used sonar array because of its cost effectiveness and low rate of interference and sensitivity to the electromagnetic field effect. As both of the localization and object detection are not completely error free in any robotic platform, we have employed mathematical approach and well defined algorithm to reduce the error and inaccuracy. For localizing the objects in the map, a trigonometric approach known as triangulation has been employed. The use of three sonar sensors in an innovative way to acquire the virtualization of twelve surrounding sonar sensors and the integration of trigonometric and geometric approaches have made it possible to localize mostly all of the objects around the robot and enabled the robot to identify the obstacles throughout the trajectory of its movement. The implementation of our method has been performed on the ground robot built using LEGO Mindstorm EV3 intelligent brick and LEJOS package have been used for programming.

Keywords—Localization, triangulation grid mapping, sonar sensor array, object detection, obstacle avoidance.

I. Introduction

Developing autonomous robot is an ever growing but intriguing task in robotics field because of the multiple interfaces, platforms, HW-SW design and unavoidable everlasting inaccuracy in the robotic instruments. As, the development of autonomous robot is targeted to enhance the public safety and tractability, an autonomous robot must able to adapt with the current environment and localization and object detection is one of the major potentiality that an autonomous robot must adopt to be adapted with the environment. Localization is the process of deciding robot's position and object detection is the process of getting information about the surrounding objects relative to robot's position. From these aspects, these two are internally related and play a fundamental and central role for the performance of an autonomous robot and any decision of action. But solving this problem is not easy because object detection and localization highly rely on input from sensors which suffer from various errors as well as the constrictive capacity of its range. Although this is true that couple of methods have been implemented to realize object detection and localization [1][2][3][4][5][6][7] but these implementation sometimes stumbles in noisy environment with lot of objects surrounding the robot and eventually many objects remain undetected. So to ensure the coverability of the area, accuracy is a difficult and challenging task to be accomplished in terms of localization and object detection. It may be reduced by using multiple sonar sensors but which eventually may not be possible due to the limit of the port, weight and shape constraints of most of the robotic platforms.

In our paper, we tried to solve this coverability issue by sonar virtualization and conveying some features including robot's freedom of rotation, simultaneous localization, imprecision in determining object's pinpoint by using sonar array along with triangulation. To ensure the precise rotation to a certain angle we have also made use of gyro sensor. The robot is able to find the objects pinpoint relative to its location based on a performance area and to find the current location, we have used the odometry data from the robots wheel. Furthermore, to make the localization more precise and for making the robot to estimate a most feasible move toward the destination, a grid map for providing static location has been built.

This paper is structured as follows. In Section II, several works, related to the object detection and localization using several methods have been introduced. Section III presents the preliminaries of triangulation. Then in section IV the environmental setup for the experiment have been presented. The presentation of the approach is described in Section V and experimental results are discussed in Section VI. At long, the conclusion and brief discussion on future work are demonstrated in Section VII.

978-1-5090-2246-5/16/$31.00 ©2016 IEEE

II. Related Works

As aforementioned, due to the enhanced inclination to the self-adaptive autonomous vehicles, dozens of researches have been done and in being progress for solving the localization, object detection and obstacle avoidance problem both in large and small scale robotics. Among them most of the tasks are solely sonar based localization without using any other third party device and with several mathematical, geometrical and statistical approach for filtering, optimizing and estimating objects pinpoint in small and large area. The preference of the sonar sensor among other technologies like IR sensors, laser sensors results from its cost-effectiveness [1][2] and comparatively more potentiality in moving object detection capability.

In [1], Bogdan uses ultrasonic sensors and shows several methods of localizing objects including the object localization by binaural sonar system where they showed the Time of Flight method for locating objects pose using single sonar sensor. Later on the object localization using multiple parallel sonar sensors have been demonstrated to show the effectiveness of object detection using more than one sonar sensors in this work.

In [3], Sooyong and Jae-Bok demonstrated the effectiveness of sonar sensor for localization by consecutive and cooperative scanning in small scale robotics mainly in bounded perimeter with edge detection.

Moreover, various mapping and filtering algorithm including Monte Carlo Localization, Kalman Filter algorithm for filtering noisy data have been introduced along with the sonar localization. In [4][5][6][7][8][15] efficiency of sonar localization has been improved by employing particle filter localization called Monte Carlo Localization. Also in [16][17][18][19] both Kalman Filter and Extended Kalman Filter methodologies have been adopted with sonar localization.

Apart from this, a couple of research works [9][10][11][12][13][14] have put together on the triangulation method with sonar localization and depicted the efficiency of this trigonometric approach. These work basically focus on getting the object's location more perfectly based on the multiple sonar sensors.

The works mentioned above have demonstrated the potency of sonar sensor based localization. In addition to these platforms, several sonar based localization works [20] [21] [22] [23] have been demonstrated in LEGO Mindstorms robotic platforms.

Sonar based localization is one of the most emphatic localization methods which can directly be derived from the previous works, mentioned in this section and the integration of our novel approach with the previous established methods assuredly contribute more to this sonar based localization.

III. Preliminaries

a. Triangulation

Triangulation is the process of deciding the location of any object from two remote points. Triangulation has a vast area of application as well as it has multiple types, including geometric triangulation, beacon triangulation, geometric circle intersections and organization of the formula for triangulation vary depending on the field of application. In terms of localization using sensors, it is the process of determining the objects location based on two sensors measurement. In this regard, the triangulated object is called landmark. In triangulation the tracked object must be in the range of two sensors. It is able to provide the coordinate and distance of objects based on geometric calculation and this geometric calculation is based on the Euclidian geometry. Besides getting the distance and coordinate of an object from a known point, the heading of a robot can also be calculated using triangulation based on the landmark detection. The formal definition of triangulation is depicted below using Fig. 3.

The basic terminology about the triangulation can be derived from Fig. 1. Here the definition of triangulation and related terminologies are provided based on the 2D Cartesian coordinate.

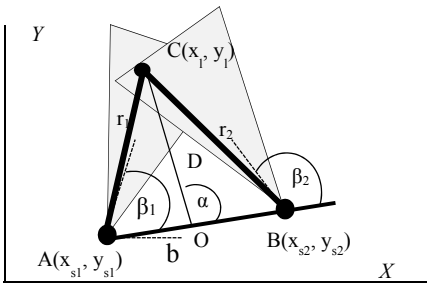


Fig 1: Triangulation using two sensor position

Assume *A* and *B* are two sonar sensors and the distance between their bearings is *b*. x_{s1}, y_{s1} and x_{s2}, y_{s2} are the known position of sensor *A* and *B* in the coordinate. α_1 and α_2 are the angles that refers to the heading of sensors *A* and *B*. Let C be any object intersected by the two sonar sensor *A* and *B*. r_1 and r_2 are the distances of the object C measured from the sensor A and sensor B consecutively. Now to calculate distance of *C* from *O* and the location of $C(x_1, y_1)$, the following rules has been adopted

$$x_1 = x_{s1} + \frac{1}{b^2}\left(d_{xs}D2 \pm |\, dys \,| \sqrt{r2^2 ds^2 - dr^4}\right) \quad \text{(i)}$$

$$y_1 = y_{s1} + \frac{1}{b^2}\left(dysD2 \pm |\, dxs \,| \sqrt{r_2^2 2\ \ ds^2 - dr^4}\right) \quad \text{(ii)}$$

Several terms mentioned in equation (i) and (ii) can be defined as follows

$$d_{xs} = x_{s1} - x_{s2} \quad \text{(iii)}$$

$$d_{ys} = y_{s1} - y_{s2} \quad \textit{(iv)}$$

$$b^2 = d_{xs}^2 + d_{ys}^2 \quad \textit{(v)}$$

$$D^2 = \frac{r_1^2 - r_2^2 - b^2}{2} \quad \textit{(vi)}$$

Now to find the distance between the object and the sensors position, the following equation needs to be applied which is the simple formulation of finding the square of distance between the two points

$$r_1^2 = (x_1 - x_{s1})^2 + (y_1 - y_{s1})^2 \quad \text{(vii)}$$

$$r_1^2 = (x_2 - x_{s2})^2 + (y_2 - y_{s2})^2 \quad \text{(viii)}$$

The equation (*i*) to (*viii*) stands for the basic notations and formulation of triangulation.

IV. Environment Setup

For the implementation of our localization approach, we have deployed a well-organized indoor environment including the software and hardware platform. As stated earlier, Lego Mindstorm EV3 intelligent brick has been used as the robotic platform which is very compact and ideal for mobile robot. LEJOS is a full-fledged package that supports Java in LEGO mindstorm systems.

A. Structure of the robot

Our experimented robot is a wheeled ground bot. we have built our robot resembling with circular shape which is 20 cm in diameter and 20 cm in height (shown in Fig. 2). The actuators are tied with three Rota caster Omni wheel which help the robot to move in any direction and at any angle very smoothly without any random displacement. The distance between any two wheels is 20cm.Three sonar sensors have been placed in 120^0 angular distance and each one's range is 250 cm. Also it has a gyro sensor for measuring the angle of rotation. Besides, the platform(LEGO Mindstorms EV3) we are using are very fast in program execution which takes only 19 seconds for localizing object and obstacles surrounding it. Altogether, the entire architecture is very convenient to localize object.

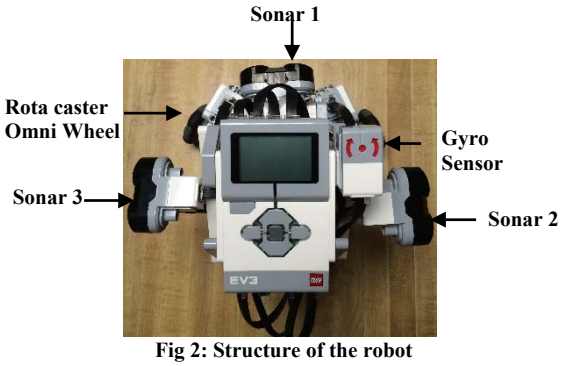


Fig 2: Structure of the robot

B. Construction of a virtual 2D static map

The experiment has been done in an indoor environment where we have created a virtual 2D grid map in a square bounding box where each cell of the grid represents a square shape with identical size. The space between the grid's nodes is 20 cm in both length and width which resembles the size of our experimented robot. The overall map is represented in 2D coordinate system (shown in Fig. 3) where each row represent y coordinate increased by 1 unit from bottom to top and each column represents x coordinate and increased by 1 unit from the left to right. The map is virtually built by LEJOS package. The code snippet for the map building is shown below in Table 1.

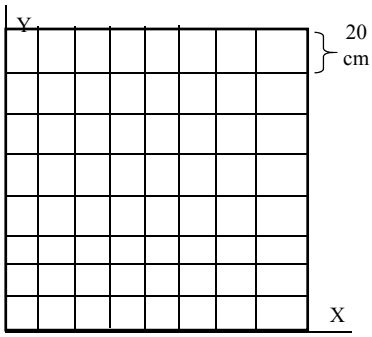


Fig 3: 2D static map

```
1. final int GRIDSPACE = 20;
2. final int CLEARANCE = 20;
3. Line [] lines = new Line[4];
4. lines.add(new Line(0, 0, 199, 0));
5. lines.add(new Line(199, 0, 199, 199));
6. lines.add(new Line(199, 199, 0, 199));
7. lines.add(new Line(0, 199, 0, 0));
8. lejos.geom.Rectangle bounds = new
Rectangle(0, 0, 200, 200);
9. this.gridMap = new LineMap(lines,
bounds);
10.        FourWayGridMesh grid = new
FourWayGridMesh(gridMap, GRIDSPACE,
CLEARENCE);
```

Table 1: Code snippet for grid map

V. Description Of Approach

In this section, the entire approach for the overall implementation has been delineated. We subdivided our overall approach into subsequent steps.

A. Localization of the robot in the map

For the detection of objects, avoidance of obstacle and keep track of the current pose of the robot, it's necessary to set up an initial pose and the heading for the robot. The initial position of the robot can be obtained from the built-in odometer of EV3 servo motor. Other than this, a gyro sensor is used to return the current heading calculation to validate the heading provided by the motor's odometer. The code snippet for localizing the robot's position is shown above in Table 2.

```
1.  public double getX()
2.  {double currentX;
3.  DifferentialPilot     dp     =          new
    DifferentialPilot(1.89f,  10.2f,  Motor.A,
    Motor.B);
4.  OdometryPoseProvider     opp     =     new
    OdometryPoseProvider(dp);
5.  currentX=opp.getPose().getX();
6.  return currentX;}
7.  public double getY(){
8.  double currentY;
9.  currentY=opp.getPose().getY();
10. return currentY;}
```

Table 2: Code snippet for finding the current position of the robot

B. Object detection and localization in the surrounding area

To be adaptive to the environment rapidly and move toward the destination without any collision, the robot needs to acquire knowledge about the objects and the obstacles of the surrounding environment by its sensing capability. As aforementioned, a trigonometric approach, triangulation is used for this purpose to enable the robot for object mapping in 2D plane.

There are three sonar sensors bind with the robot and the robot will rotate itself by 90^0 in three 30^0 rotations and with a defined interval at the end of every 30^0 for better sensing of the environment. As those three sensors S1, S2 and S3 are placed at an interval of 120^0 from one another (shown in Fig. 4), after a 90^0 rotation, the robot will have distance values of 12 sensors around 360^0 view using its three sonar sensors (Fig. 5) and will be able to get all the objects and the obstacles surrounding it as well as in its path.

The distance values after every rotation of each sensor is fetched and stored. The virtual position of 12 sensors is shown in Fig. 6

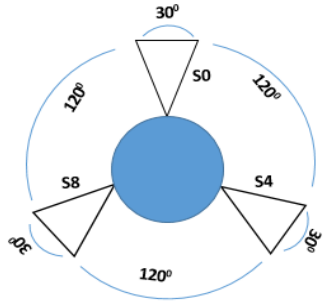
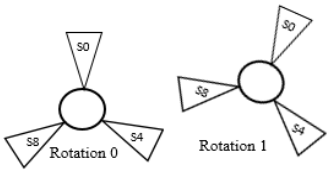


Fig 4: Sensors position with an angle distance of 120^0



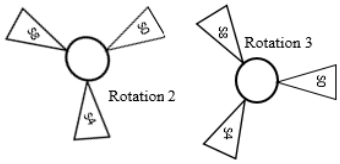


Fig 5: Sonar position after every 30^0 rotation

The process of mapping an environment and finding the pinpoint (2D coordinate) of an object in the map by detecting object's distance from the robot's relative position is quite impossible with only one sonar sensor value. Because in noisy environment, there may have multiple objects through the same X axis or in the range of the one sonar wave (Fig. 7) of the map which causes the sonar sensor to provide same distance values for all of the visible objects in the same X axis which will eventually make the precise object detection in the map impossible. Use of multiple sensors ray de-escalate this issue. In this regard, two adjacent sonar sensors have been considered to detect an object precisely in the 2D map. As we have 12 sonar sensors data after 90^0 rotation, to have the values available, we have stored all the values in an array.

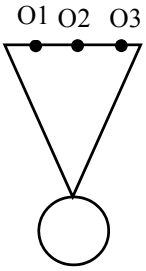


Fig 7: multiple objects in the range of one sonar wave

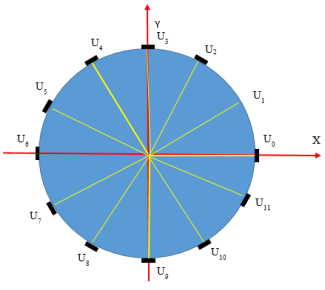


Fig 6: Position of 12 virtual sonar sensors

Let's consider the following array k_n where the size of k is 12 which represents the total number of memory slots in the array k and n = 0, 1, 2, 3 ……. 11. After every rotation the sonar values are stored in three memory cell of that array cell for the convenience of data manipulation. The order of storing the sensor values based on Fig. 6 is shown in Table 3.

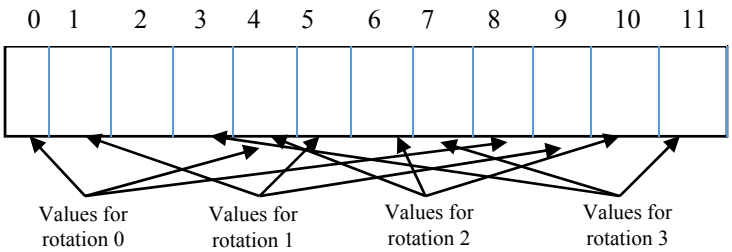


Fig 8: Representation of array for storing sonar values

Rotation	Sensor #	Array Cell
Rotation 0	0,4,8	k_0, k_4 k_8
Rotation 1	1,5,9	K_1, k_5 k_9
Rotation 2	2,6,10	K_2, k_6 k_{10}
Rotation 3	3,7,11	K_3, k_7, k_{11}

Table 3: Order of storing values in each rotation

C. Saving sensors values into a defined array

If *s* denotes the sensors, *r* represents the rotation angle (30^0), *n* is the memory segment as mentioned earlier, and *d* is the distance value from each sensor and after every rotation the values from the sonar sensors are stored in distinct segments of the memory for further processing and the selection of the successive memory can be depicted as follows

$$\{\forall r_i\, \forall s_j: d(s_j) \in k_n \text{ where } n=\begin{cases} k-(k-(1\times i)) \\ k-(k-(4+(1\times i))) \\ k-(k-(8+(1\times i))) \end{cases} \textit{where}$$

$\{\forall r_i: \ 0 \le i \le 3\}$, $\{\forall s_j: \ 1 \le j \le 3\}$, and $k=12$

After every rotation the three sensor values are to be stored in the designated array location (shown in Fig. 10) simultaneously. It is noteworthy that, *i* starts from 0 and r_0 denotes there is no rotation which indicates the initial position, and *i* is increased by 1 after every 30^0 rotation. The code snippet for storing the values in particular array cell is presented in Table 4.

```
1. for(int i=0;i<=3;i++){
2. n = n- (n-(1*i));
3. if(n==0)  {
4. distanceStorage[n]=sample1[0]*100;
5. distanceStorage[n+4]=sample2[0]*100;
6. distanceStorage[n+8]=sample3[0]*100;
7. gyro.resetSensor();  }
8.   if(n>0){ while(gyro.getAngle()!=30) {
9.     Motor.A.setSpeed(100);
10.    Motor.B.setSpeed(100);
11.    Motor.A.backward();
12.    Motor.B.forward(); }
13.    System.out.println(gyro.getAngle());
14.    Sound.beep();
15.    distanceStorage[n]=sample1[0]*100;
16.    distanceStorage[n+1]=sample2[0]*100;
17.    distanceStorage[n+2]=sample3[0]*100;
18.    gyro.resetSensor();}
```

Table 4: Code snippet for storing values into the array

D. Making combination of two sensors for triangulation

After a complete 90^0 rotation, each object's relative distance is calculated using triangulation by taking two values from the two consecutive location of the array. For generalization, the combination of all 12 sensors can be represented by the following:
$\{\forall i: \; s_{s\text{-}i} \bigwedge S_{s\text{-}|(i-1)|} \; for\ all\ 0 \le i \le 11\ \}$ where s denotes the sensors where s=12 and $s_{s\text{-}i}$ represents the first sensor in the combination and $S_{s\text{-}|(i-1)|}$ represents the second sensor. The code snippet for the combination of two adjacent sonar is shown in Table 5.

```
1. for(int i=0;i<12;i++){
2. if(i<11 && distanceStorage[i]<=mapRange){
3. d1 = distanceStorage[i];
4. d2 = distanceStorage[i+1]; }
5. If
(i==11&&distanceStorage[i]<=mapRange) {
6. d1 = distanceStorage[i];
7. d2 = distanceStorage[i-10];}
```

Table 5: Code snippet for combining two adjacent sonar rays for triangulation

E. Implementation of triangulation

Immediately after having all the combination and data from the sonar sensors, the triangulation method has been applied to find the distance of the objects from the robot's relative position in the map. As mentioned earlier, the values from the two adjacent ultrasonic sensors are considered for triangulation, we have to consider the distance between the two ultrasonic sensors and hence calculated as follows, distance between two sonar sensor, b = 12 cm (depending on the angle separation of angle 30^0). Here, we have calculated the distance between the robot and the point in terms of point p to locate the object directly relative to the robot (shown in Fig. 9). For the triangulation, all the equations are derived keeping consistent with the equations from (i) to (viii).

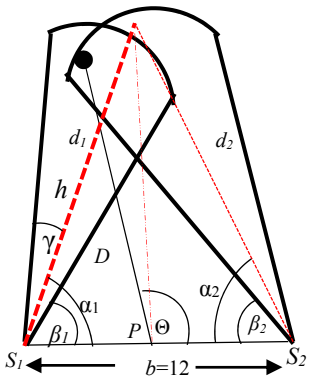


Fig 9: Triangulation based on two sonar waves

For the triangulation we have to find out the $\beta 1$ *and* $\beta 2$ which is the angle value in the range of detection field.

$$cos\ \beta_1 = \frac{b^2 + {d_1}^2 - {d_2}^2}{2bd_1} \qquad \text{(ix)}$$

Similarly if consider the detection range of the another sensor then,

$$cos\ \beta_2 = \frac{b^2 + {d_2}^2 - {d_1}^2}{2bd_2} \qquad \text{(x)}$$

where d_1 and d_2 refer to the distance values from the combination of two sonar and second sensors for each combination for triangulation.

F. Determining the intersected objects for triangulation

With the virtualization of 12 sonar sensor using only 3 sensors, it is possible to get all of the objects around the robot after 90^0 rotation of the robot in either left or right. But all the objects may not be in the range of the intersection or in the detection area of two adjacent sensors which will eventually results in the faulty localization of the objects. That's why, its needs to be ensured that objects are in the detection area of two adjacent sonar waves.

From the Fig. 9, β_1 and β_2 are the orientation of two sonars in the detection angle, γ is the half of the angle of detection range of the sonar wave and α_i (i=1, 2) is the angular orientation of both sonar sensors. Any objects will be in the intersection or in the detection range of both waves if it is detected maintaining the angle α_i and γ. Hence, for the objects to be triangulated, the angle of detection range for two adjacent sonar must satisfy the following two conditions

a. $\cos(\alpha_1 - \gamma) \le cos\beta_1 \le \cos(\alpha_1 + \gamma)$
b. $\cos(\alpha_2 - \gamma) \le cos\beta_2 \le \cos(\alpha_2 + \gamma)$

It is noteworthy that, α_i and γ are constant and α_1 and α_2 are equal. For our system $\gamma = \frac{\text{detection angle of the sonar}}{2}$ and for the calculation of α we have adopted the following trigonometric equations

$$h = \sqrt{\frac{b^2}{4} + w^2} \qquad \text{(xi)}$$

$$\alpha = \cos^{-1}\frac{b}{2} + h \qquad \text{(xii)}$$

To meet the conditions a and b, the distance D from point P and Θ is calculated as

$$D = \sqrt{{d_1}^2 + \frac{b^2}{4} + d_1 b cos\beta 1} \qquad \text{(xiii)}$$

Then the angle α is measured to calculate the deviation of the object from the straight heading of the robot.

$$\Theta = \cos^{-1}\frac{d1^2 - \frac{b^2}{4} - D^2}{bD} \qquad \text{(xiv)}$$

The code snippet for the above process is depicted below in Table 6

```
1.      cosB1=((Math.pow(b, 2)+ Math.pow(d1, 2)-
  Math.pow(d2,2)) /(2*b*d1));
2.      cosB2=((Math.pow(b, 2)+ Math.pow(d2, 2)-
  Math.pow(d1, 2))/(2*b*d2));
3.      B1=Math.acos(cosB1)*(180/Math.PI);
4.      B2=Math.acos(cosB2)*(180/Math.PI);
5.      limit1 = Math.cos(alpha-gamma)*
  (180/Math.PI);
6.      limit2 = Math.cos(alpha+gamma)*
  (180/Math.PI);
7.      if(B1>=limit1 && B1<=limit2 && B2>=limit1
  && B2<=limit2){
8.      D=  Math.sqrt((Math.pow(d1,
  2)+((Math.pow(b, 2)/4)) + (d1*cosB1)));
9.      System.out.println("D:"+D);
  theta = (Math.acos((Math.pow(d1, 2)-((Math.pow(b,
  2)/4)))-(Math.pow(D, 2))) * 180/Math.PI)/(b*D); }
```

Table 6: Code snippet for combining two adjacent sonar rays for triangulation

It is noteworthy that, whenever an object is detected by any of the combination of two adjacent sensors then, that

combination is marked along with its positional and orientation angle for further processing to find out the exact coordination of the object in the map.

G. Localizing the object inside the 2D map

Finally, localizing the objects in the predefined map by finding its Cartesian coordinate in 2D plane, is the prerequisite for the robot to avoid it if it is in front of its path of movement. As our robots gets all the objects around it, the location of the

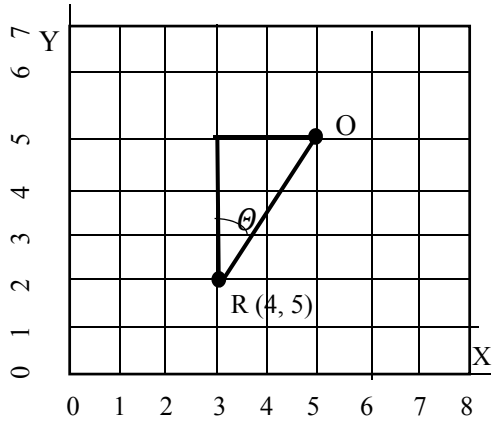


Fig 10: Localization of object inside the map

objects may be in any of the directions, not necessarily only in the left or right quarter of the coordinate or just in front of the robot.

At this stage as we have the distance and the angle values from the position of the robot to the objects, finding out the coordinate of the objects in the map is now a trivial task. As before, we will make use of the triangulation for finding out the coordinate of the object in the map.

As aforementioned, the distance between the two adjacent nodes of the grid in our map is 20 cm. With this in consideration and to make the localization exact, we have fixed the distance between the two adjacent nodes 20 cm to avoid collision with one object to another. That is, the travel distance of the robot in either x or y direction is divided by 20 to get the x or y location in the map which also true for distance *D* measured from sonar sensor.

Now, as mentioned earlier, we have the robot current pose (X_R, Y_R), distance *D* and the angle Θ. So the object coordinate O (shown in Fig 10) in the map has been calculated as follows

$$O(x, y) = (X_R + cos\Theta \times D, Y_R + sin\Theta \times D) \quad \text{(xv)}$$

But this is true for the process of detecting objects in unidirectional mode but for our sonar virtualization, objects in different directions are being detected simultaneously by 12 virtual sensors at different angles which nullifies the equation (xv) for providing valid coordination of the objects. Therefore, to get the valid coordination, we have employed the following procedure.

As stated early, each combination of two adjacent sensor has a constant positional angle and when it measures the distance of the objects from the position *p* (as in Fig. 9) the angle value of the combination of two adjacent sensors is 60^0 and as we have rotate our robot from right to left for the gyro sensor to provide positive value, the positional angle value will be incremented by 60^0 for the combination from C1 to C6 consecutively relative to the heading and current coordination of the robot. So the modified equation is as follows

$$O(x, y) = x_R + \cos(\Theta_{Cn} + \Theta) \times D, y_R + \sin(\Theta_{Cn} + \Theta) \times D \quad \text{(xvi)}$$

This is how we can get the x and y coordinate of the object relative to the robot's position in the map.

VI. EXPERIMENTAL RESULTS

The experimental study was conducted in an indoor environment which is larger than the grid map defined beforehand and the map is programmed using the LEJOS package (shown in Table 1). Considering the range of the sonar sensor and size of the robot aforementioned, the perimeter of the map has been programmed which is 199×199. Initial gyro value is set to 0^0 and the object detection has been performed by setting different initial pose of the robot. We have also vary the shape and the number of objects around the robot to see the result. Since, the "gridSpace" and the "gridClearence" is 20cm in our program (shown in Table 1) to represent 20 cm as 1 unit or 1 node count in the map. Hence, after each reading fetched from the odometer as well as from the sonar sensors, the reading is divided by 20cm to get the exact X and Y coordinate in the map to determine which node is occupied and which is not. We have executed our experiment several times. The result from one of our experiment is shown in Table 7.

Object #	Sonar sensors pair	Distance D (cm)	Position (cm)	Coordinate
1	S0 & S1	23.36	(60,40)	(3,2)
2	S2 & S3	117.88	(20,40)	(1,2)
3	S4 & S5	124.35	(100,40)	(5,2)
4	S6 & S7	38.82	(40,80)	(2,4)
5	S9 & S10	59.16	(60,40)	(3,2)

Table 7: Data collected from the experimental result

We have developed "Dis-Centralized Heterogeneous Robot Control System" to fetch the values from the robot [25]. Fig. 11 shows the experimental values fetched by our "Heterogeneous Robot Control System". First we put the robot in the map then set the initial pose (x, y, Θ) from the "Initial Pose" panel where Θ is the current heading. We have changed the position of the robot from the "Control" panel. Panel "Sonar Reading" shows the data from the sonar sensors. When any of the 12 sensors (Virtual) failed to get any object inside the perimeter of the map, it returns "NaN". Panel "Intersection" shows the possible objects in the intersection of two sonar wave which is triangulated. Moreover, to avoid the detection of unwanted fixed the range of sonar to 200cm from the "Sonar Range" panel.

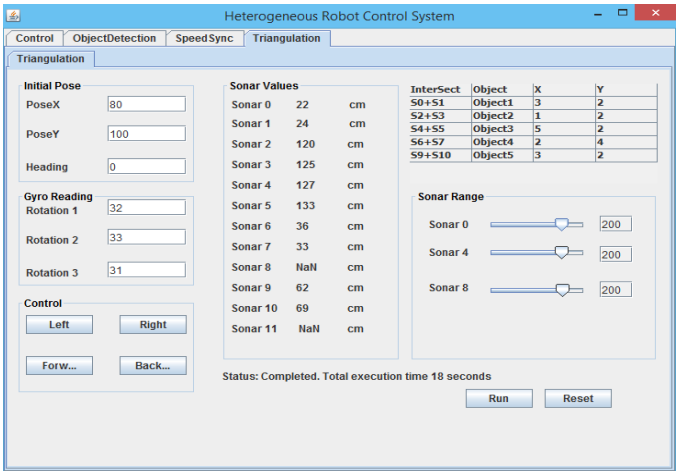


Fig 11: Values fetched Robot Control System

For this, experiment shown in the Fig. 12, we have put the object around the robot as follows

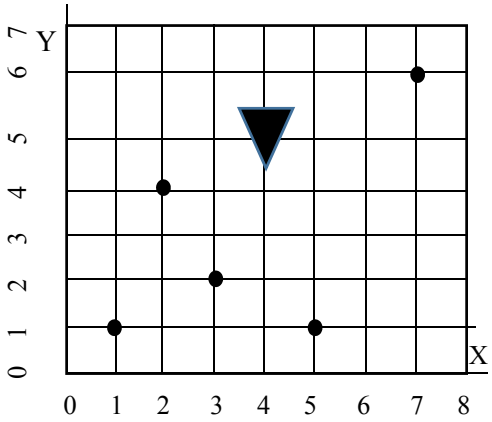


Fig 12: Localization of object inside the map

Here dot denotes the objects and the triangle refers to the robot. The real scenario of the above figure is as follows where we used some round objects as landmarks.

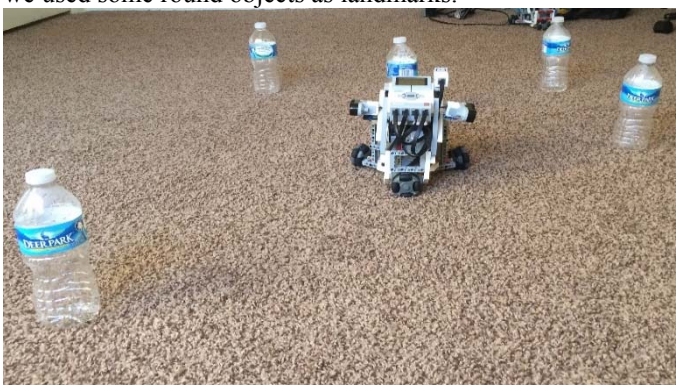


Fig 13: Scenario of the localization environment

We have relied on the odometry data, sonar sensor and gyro sensor for this object detection and localization. As no equipment among them is free from error and drift hence, some of the objects locations are overlapped or even we got the correct position for X but for Y, it's slightly deviated from its own co-ordinate. From the Fig. 13 in the "Gyro Reading" panel we can see the gyro and odometer drift. We set the rotation 30^0 but for the gyro and odometer drift we got slightly different data. Our robot rotates 96^0 rather than 90^0 which is obvious from the control system. Besides after the calculation we suffered from the floating point error because of different trigonometric and geometric formulas. But even after this, we are able to detect all the objects alone with their X and Y coordinate (shown in Table 7) around the robot using this implementation which utter the potentiality of this novel method.

VII. Conclusion and Future Work

Our study and experiment have been done with the aspiration to solve the errant localization and object detection problem efficiently by adopting a hybrid organization of sonar sensor array for getting the best view of the objects and the obstacles by the autonomous robot. Basically, this study of localization and object detection has been accomplished solely based on the sonar sensor array and upgraded trigonometric formulation where the sonar sensors have been attempted to be utilized in a great extant by resolving the technical issue of interference among the physical sonar sensors with a view to increasing the precision of object and obstacle detection. Besides, the presented customized algorithms for the object localization demonstrated an extensive study with a significant result.

Our future work on this localization is to be done for the purpose of making this method more robust by employing different well established mathematical approaches. As mentioned in the experimental result about the drift of sensors that have made the localization for some objects erroneousness, So the fundamental focus for our future work will be minimize these error. For reducing the sensors drift, we will use the Kalman filter. Moreover, to make the best possible move and to determine the occupancy of any specific position in the map, the Occupancy Grid Mapping algorithm will be used.

For both small and large scale robotics, it is crucial to generate the possible shortest path for the robot from origin to the destination and keeping this in mind, we will use the A* search algorithm for path generation based on the obstacle detection.

Apparently, this consistent interaction among the sonar sensors stands for the powerful capability of sonar localization that will act as a gateway for the further research in both large and small scale sonar based localization in the field of robotics.

In addition to this, this research study is validated on a current up-to-date educational robotics platform EV3 using LEJOS package which provides a profound and an extensive impact on the object oriented programming and design study as well as significant effects on the broader educational aspects.

Acknowledgment

This project is supported by Air Force Research Lab under award no of FA87501520106. We would like to thank all reviewers for the valuable inputs.

REFERENCES

[1] Bogdan Kreczmer, "Object Localization and Differetiation Using Ultrasonic Sensors." (2010). INTECH Open Access Publish.

[2] Vassilis Varveropoulos, "Robot Localization and Map Construction" Using Sonar Data," The Rossum Project.

[3] Baek, S.,Park, H., & Lee, S.."Mobile robot licalization based on consecutive range sensor scanning and optical flow measurements." ICAR '05. In Proceeedings of 12th International Conference on Advanceed Robotics, 2005

[4] Muller, J., Rottmann, A., Reindl, L., & Burgard, W. "A probabilistic sonar sensor model for robust localization of a small-size blimp in indoor environments using a particle filter." 2009 IEEE International Conference on Robotics and Automation.

[5] Thrun, S., Fox, D., Burgard, W., & Dellaert, F. "Robust Monte Carlo localization for mobile robots". Artificial Intelligence, pp. 99-141.

[6] Müller, J., Gonsior, C., & Burgard, W. "Improved Monte Carlo localization of autonomous robots through simultaneous estimation of motion model parameters." 2010 IEEE International Conference on Robotics and Automation.

[7] Thrun, S., Fox, D., Burgard, W., & Dellaert, F. "Robust Monte Carlo localization for mobile robots." Artificial Intelligence, pp. 99-141.

[8] Theodoridis, T., Hu, H., Mcdonald-Maier, K., & Gu, D. "Kinect Enabled Monte Carlo Localisation for a Robotic Wheelchair." Advances in Intelligent Systems and Computing Intelligent Autonomous Systems 12, pp. 153-163.

[9] Wijk, O., & Christensen, H. "Triangulation-based fusion of sonar data with application in robot pose tracking." IEEE Trans. Robot. Automat. IEEE Transactions on Robotics and Automation, pp. 740-752.

[10] Rekleitis, I., Dudek, G., & Milios, E. "Experiments in free-space triangulation using cooperative localization," in Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453).

[11] Tekdas, O., & Isler, V. "Sensor Placement for Triangulation-Based Localization." IEEE Transactions on Automation Science and Engineering IEEE Trans. Automat. Sci. Eng., pp. 681-685.

[12] Esteves, J., Carvalho, A., & Couto, C. "Generalized geometric triangulation algorithm for mobile robot absolute self-localization." 2003 IEEE International Symposium on Industrial Electronics (Cat. No.03TH8692).

[13] Vicente, J., Sales, J., Marin, R., & Sanz, P. "Multi-Sensor Localization and Navigation for Remote Manipulation in Smoky Areas." Int J Adv Robotic Sy International Journal of Advanced Robotic Systems, 1-1.

[14] Melo, L., Rosário, J., & Junior, A. " Mobile Robot Indoor Autonomous Navigation with Position Estimation Using RF Signal Triangulation." POS Positioning, pp. 20-35.

[15] Huang, J. "Robot Position Identification by Actively Localizing Sound Beacons." 2006 IEEE Instrumentation and Measurement Technology Conference Proceedings.

[16] Rudy Negenborn, "Robot Localization and Kalman Filters On finding your position in a noisy world." M.S. thesis, Dept. Computer Science., Utrecht University. Netherlands, 2003.

[17] Quinlan, M., & Middleton, R. "Multiple Model Kalman Filters: A Localization Technique for RoboCup Soccer." RoboCup 2009: Robot Soccer World Cup XIII Lecture Notes in Computer Science, pp. 276-287.

[18] D'alfonso, L., Griffo, A., Muraca, P., & Pugliese, P. "A SLAM algorithm for indoor mobile robot localization using an Extended Kalman filter and a segment based environment mapping." 2013 16th International Conference on Advanced Robotics (ICAR).

[19] Cotugno, G., D'alfonso, L., Lucia, W., Muraca, P., & Pugliese, P. " Extended and Unscented Kalman Filters for mobile robot localization and environment reconstruction." 21st Mediterranean Conference on Control and Automation.

[20] Lee, J., & Buitrago, J. " Map generation and localization for a LEGO NXT robot." 2015 IEEE 2nd Colombian Conference on Automatic Control (CCAC).

[21] Pinto, M., Moreira, A., & Matos, A. " Localization of Mobile Robots Using an Extended Kalman Filter in a LEGO NXT." IEEE Trans. Educ. IEEE Transactions on Education, pp. 135-144.

[22] Kicman, P., Silson, P., Tsourdos, A., & Savvaris, A. (2011). "Cartographer - A Terrain Mapping and Landmark-based Localization System for Small Robots." Infotech@Aerospace 2011.

[23] Lee, J., & Buitrago, J. "Map generation and localization for a LEGO NXT robot". 2015 IEEE 2nd Colombian Conference on Automatic Control (CCAC).

[24] Wijk, O., & Christensen, H. "Triangulation-based fusion of sonar data with application in robot pose tracking." IEEE Trans. Robot. Automat. IEEE Transactions on Robotics and Automation, pp. 740-752.

[25] M.H. Shuvo, "Trusted distributed centralized control architecture for heterogeneous robotic system," submitted at ICSE-SRC, May 14-22, 2016.