RESEARCH ARTICLE

Jesús Antonio Álvarez Cedillo, Elizabeth Acosta Gonzaga, Juan Carlos Herrera Lozada, Teodoro Álvarez Sánchez, Jacobo Sandoval Gutiérrez, Mario Aguilar Fernández

MECHANICS
Applied robotics

PARTICLE FILTER BASED LOCALIZATION FOR A MOBILE ROBOT

[1]Jesús Antonio Álvarez-Cedillo,[1]Elizabeth Acosta-Gonzaga,[2]Juan Carlos Herrera-Lozada, [3]Teodoro Álvarez-Sánchez, [4]Jacobo Sandoval-Gutiérrez, [1]Mario Aguilar-Fernández,

[1]Instituto Politécnico Nacional, UPIICSA, Av. Té 950, Iztacalco, Granjas México, 08400 Ciudad de México, México, Teléfono: (+52) 55 5729 6000 jaalvarez@ipn.mx, eacostag@ipn.mx, maguilarfer@ipn.mx
[2]Instituto Politécnico Nacional, CIDETEC, Av. Juan de Dios Bátiz s/n, Gustavo A. Madero, Nueva Industrial Vallejo, 07700 Ciudad de México, México, Teléfono: (+52) 55 5729 6000, extensión 52527, jlozada@ipn.mx
[3]Instituto Politécnico Nacional, CITEDI, Av. Instituto Politécnico Nacional 1310, Nueva Tijuana, Tijuana, Baja California, México, talvarez@citedi.mx
[4]UAM Lerma, Av. Hidalgo Pte. 46, La Estación, 52006, Lerma de Villada, Estado de México, México, jacobosandovalg@gmail.com

Received: 24/Nov/2016—Reviewed: 28/Sep/2017--Accepted:17/Oct/2017--DOI: http://dx.doi.org/10.6036/NT8222

TO CITE THIS ARTICLE:
ALVAREZ-CEDILLO, Jesus Antonio, ACOSTA-GONZAGA, Elizabeth, HERRERA-LOZADA, Juan Carlos et al. PARTICLE FILTER BASED LOCALIZATION FOR A MOBILE ROBOT. *DYNA New Technologies*, Enero-Diciembre 2017, vol. 4, no. 1, p.[19 p.]. DOI: http://dx.doi.org/10.6036/NT8222

ABSTRACT:
The field of mobile robotics has been of great interest in recent decades; therefore, a variety of techniques has been proposed for its development. This article presents an algorithm for the navigation of mobile robots, using the location of particle filters, the Monte Carlo method, which uses the sampling technique to divide the state space into samples and thereby to increase the number of samples to discretion. To validate the proposed algorithm a mobile robot was constructed using ultrasonic sensors and a modified version of Dieter's algorithm. The results show that the proposed algorithm is computationally efficient with low complexity, easy to implement and produces accurate and efficient outputs. These findings may be useful for the development of robotic applications where position is critical, and accuracy is important. This proposal can be very useful in industrial applications.

Keywords: The Monte Carlo Localization, mobile robot, navigation, low complexity algorithms.

RESUMEN:
El campo de la robótica móvil ha sido de gran interés en las últimas décadas, por lo que, se han propuesto una diversidad de técnicas para su desarrollo. Este artículo presenta un algoritmo para la navegación de robots móviles, utilizando la localización de filtros de partículas, el método de Monte Carlo, el cual utiliza la técnica de muestreo para dividir el espacio de estado en muestras y con ello, aumentar el número de muestras a discreción. Para validar el algoritmo propuesto se construyó un robot móvil usando sensores ultrasónicos y una versión modificada del algoritmo de Dieter. Los resultados muestran que el algoritmo propuesto es computacionalmente eficiente con baja complejidad, fácil de implementar y que produce salidas precisas y eficientes. Estos hallazgos pueden ser útiles para el desarrollo de aplicaciones robóticas donde la posición es crítica, y la precisión es importante. Esta propuesta puede ser muy útil en aplicaciones industriales.

Palabras clave: Localización Monte Carlo, robot móvil, navegación, algoritmos de baja complejidad.

1.- INTRODUCTION

Autonomous mobile robots are an active area of research because it is necessary to give a task that the robot must follow. A mobile robot should be an intelligent, autonomous machine. In general, this robot is used in various applications in offices, hospitals, museums, homes, human environments and war camps.

The literature review on mobile robot navigation suggests, in relation with the number of publications, contributions in areas such as artificial intelligence [1], evolutionary algorithms [2, 3, 4, 5, 6, 7, 8, 9], and others algorithms [10, 11, 12, 13, 14, 15, 16, 17], between the most relevant.

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

Mobile robots can have others features such as WLAN (Wireless Local Area Network), teleoperation control. These robots can work for us like a robot maid, or they can play with us as a smart home mate. The robot is expected to have human-like motion capacity and must have the environmental perception to cope with the real world. However, implementing autonomous perception and navigation in a mobile robot is a challenging task [18, 35].

Thus, if the possibility exists that the robot is to navigate in a human environment then sensing devices such as cameras, ultrasonic sensors, and laser**,** are necessary for map building of mobile robots. Several works show techniques of multiple sensor fusion [17, 10]. Other authors have published papers where the focus is to enhance the map building capability of mobile robots [4, 6, 12, 14, 15].

Recently, researchers have been interested in putting several robots to work collaboratively, and merge perceived environmental information [19; 20; 21]. Compared with a single robot and multiple sensing positions, multiple robot systems provide more reliable and efficient solutions. Each robot can be made simpler with possibly only one sensing device. Multi-robot systems have been developed with heterogeneous or homogeneous robots collaborating in map building. The map-building process will become faster and more efficient with cooperative multiple mobile robots.

In the proposed architecture, the local sensory data from each ultrasound sensor is recorded; the information of different locations are fused to obtain a global map, which will be analyzed using Kalman filtering techniques [23].

Since each robot is connected to the network through WLAN, a server is responsible for sensory data fusion and producing an integrated representation. An Individual robot is used for navigation and task execution. Robot interconnection using WLAN offers great convenience for visual information collection and distribution [22].

1.1 POSITION ESTIMATION FILTERS

In the literature are three main approaches to estimate the position of a robot, and these procedures are called filters, each filter depends on kind of problem and resource employed, these filters are:

1. Kalman Filter
2. Markov localization
3. Monte Carlo Localization

The Kalman filter provides a means of combining measurements and knowledge of a processor system to determine the value of some desired quantity [6]. To illustrate this, consider the following example. A researcher selects a resistor that being 100 ohms is labeled with a tolerance of 1%. To ensure that the resistance is 100 ohms, the researcher measures the resistance with an ohmmeter that has an accuracy of 10% [24].

The resistance the researcher measures is 105 ohms. What value of resistance should the researcher assume for the resistor? The resistor is known to be accurate within 1% and the ohmmeter within 10%. Therefore, it seems logical to lean toward the value of 100 ohms. However, the ohmmeter read 105 ohms. Therefore, the researcher may wish to account for this and specify a resistance of 100.5 or 101 ohms. Thus, in determining the strength, the researcher has considered the known information about the resistor and the measurement.

The researcher gave more weight to the specified resistance than to the ohmmeter measurement because of the relative accuracies involved. In this sample, the tolerance on the resistor was 20%, more or less. The experimenter was inclined to assign a value close to 105 ohms to the resistor.

In essence, the above decision processes are performed by the Kalman filter. By accounting for a process, or system, and measurement uncertainties, the Kalman filter estimated quantities based on measurements and expected behavior of the scheme.

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

Other researchers have found the usefulness of the Kalman filter, for example, [32], made a comparison of estimation algorithms for mobile robot navigation; they found that, Multiple model Kalman Filter is the best estimation algorithm.

The Markov localization is used to solve the problem of state estimation from sensor values. It is a probabilistic algorithm. The Markov localization maintains a probability distribution over the space of all different positions [29].

The Markov algorithm represents the relative motion since the last update and the current ultrasonic sensor input. The algorithm has two procedures: A) Motion update which changes the position to reflect the known motion since the last update, and it refreshes the weights to the new sensor ultrasonic input. It gives the relative movement to each sample location. B) Resampling is done when the samples contribute little to the distribution. Little weight samples are deleted.

The original algorithm has the following characteristics:

1. Resampling is done only on little weight samples.
2. The distribution is initialized near of the map.
3. Dither is added when resampling.
4. In resampling, the prior weight of the new sample is a fraction of the base sample weight.

The Markov localization has been used in applications such as [33], who present an intellectual method of guiding mobile robot navigation using reinforcement learning algorithm. The Markov decision process is used to improve the performance of the robot navigation.

Monte Carlo Localization (MCL) or Particle Filtering represents a new approach to the problem of robot localization estimating a robot's location in a public environment, given its movements and sensor reading over time.

In order to understand The Monte Carlo localization supposes a floor with three doors, two of which are close together, and the third further down the corridor. Assume that this robot can detect entries and can tell whether it is in front of a wall or in front of a door. Such a feature can serve the robot as a landmark. Given a map of this pure environment and no information whatsoever where our robot is located, it is possible to use it as a marker to drastically reduce the space of possible locations [25].

It is feasible to describe the robot’s position with three Gaussian distributions; each distribution centered in front of a door and its variance a function of the uncertainty with which the robot can detect a door’s center. From error propagation law are has:

The Gaussians describing the robot’s three possible locations allow moving the robot, the variance of each Gaussians allow increasing with the distance when the robot moves.

Given a map of the environment, the map generated by three Gaussian distributions is now feasible and the location of the three doors; the doors are not equally spaced, and only some of the peaks will coincide with the site of entry [26, 27, 28].

In this paper a Monte Carlo filter localization variant was used, Kalman filter and Markov localization were employed to compare the error, possible implementation, and computational cost.

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

2.- METHOD

In this paper the global localization problem is resolved; in this case, the robot can't know the starting position but needs to figure out where it is. To make things a bit simpler, it is necessary to solve this problem in a one-dimensional world (because each ultrasonic sensor detects an object in one frontal line). The primary Monte Carlo Localization is shown as follows:

Suppose a long hallway with a set of open entries along one side. The doors are distributed unevenly along it, and the doors are not all of the same widths. The robot moves back and forth along the hallway, and at any time is at the door or in a wall segment.

The robot's task is a move to a predefined goal point along the hallway. But it doesn't know its starting point. It does have a sonar unit that is aimed at the wall or doors. This unit is reliable, and must be used to determine if the robot is currently in front of a door or a wall. To solve this problem, the robot has to move forward a fixed distance and take a sonar reading. If it reaches the end of the hallway, it starts backing up instead of moving forward. Sometimes, the robot needs to go back and forth along the hallway a few times before it accurately knows where it is.

3.- IMPLEMENTATION

The mobile robot proposed for this project must follow:

1. It can reliably move back and forth along a linear path.
2. It can reliably move a certain distance.
3. It has an ultrasonic sensor unit aimed perpendicularly to the axis of movement.
4. It can receive a command from a base computer or tele-command.

The platform proposed is a differential wheel robot controlled by Arduino as is shown in Figure 1. They show the component distribution and the position of the three ping sensors. Also, the mobile robot has an odometer such as an auxiliary device in order to have a better measure of the distance.

1. The size is 0.10 x 0.10 x 0.04 m
2. The mechanical design, it was designed using Blender open source software.
3. All physical structure were builded by using a 3D printer, with ABS plastic.

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

Jesús Antonio Álvarez Cedillo, Elizabeth Acosta Gonzaga, Juan Carlos Herrera Lozada, Teodoro Álvarez Sánchez, Jacobo Sandoval Gutiérrez, Mario Aguilar Fernández

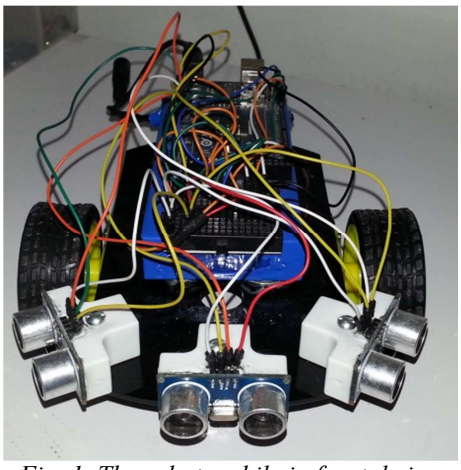

Fig. 1. The robot mobile in frontal view

The finished prototype has a 12 volts LIPO battery and two Pololu's motors 1:50 relationship .

The Monte Carlo Localization is based on a collection of particles or samples. Each particle consists of a possible location the robot may currently occupy, along with a value which represents importance weights. More samples are necessary with the aim that the program can converge to a correct solution; but computational cost is high. Thus, the use of more resources affecting the performance is necessary.

When the program starts, the robot does not know where it is. However, the current samples are evenly distributed, and the importance weights are all the same. After much time, samples near the actual current position are more likely, and those further away less likely. The basic algorithm is as follows:

1. Initialize the set of samples; their locations are evenly distributed and have the same weights.
2. Repeat for each sample:
   a) Move the robot a fixed distance and read the sensor.
   b) For each particle update the location.
   c) Assign the importance weights of each particle to the likelihood of that sensor reading given that new location.
3. Create a collection of samples by sampling with replacement from the current set of samples based on their importance weights.
4. Let the group become the current round of samples.

In step four, samples with a high importance weight are more likely to be chosen than those with low probability.

However, the proposed algorithm is similar to The MCL algorithm proposed by Dieter Fox [30, 31]. For generating at time t the next set of samples $St + 1$ from the current set St. See Listing 1.

The main differences are: the xt variable is the location and the wt the probabilities with weight, so the *xt*, *wt* couple represents a sample. The distance traveled is it, and the sensor reading is it, so *xt(i)* is the location of a sample at time t and n is the number of samples. This behavior is shown in Listing 2.

The Monte Carlo Localization algorithm proposed is used to estimate the position and orientation of a robot in its environment using a known map using ultrasound ping sensor data and an odometer sensor data. Thus, to localize the robot, the algorithm uses the particle filter algorithm to estimate the position. The particles represent the distribution of

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

the states for the robot, then, each particle knows the robot state. It is very important to calculate the near zone and the beam spread of the Ping sensor 28015 of PARALAX and with this features it is possible having a better control on the obstacles.

The commercial features reported by PARALAX are:

1. Range: 2cm to 3m (0.8 in to 3.3 yd).
2. Burst indicator LED shows sensor activity.
3. Bidirectional TTL pulse interface on a single I/O pin can communicate with 5 V TTL or 3.3 V CMOS microcontrollers.
4. Input trigger: positive TTL pulse, 2 µs min, 5 µs typ.
5. Echo pulse: positive TTL pulse, 115 µs minimum to 18.5 ms maximum.
6. RoHS Compliant.

The frequency is .40 Mhz, velocity of 6400 m/s, and diameter of 10 mm and 6 dB spread.

Wherever, to calculate the Near Zone it is used the next expression:

$$Near\ Zone = \frac{D^2}{4\lambda} = \frac{100}{4} = 1.6\ mm$$

K in a beam spread of 6 dB value 0.51. Using the next expression, it is possible obtain the angle:

$$Sin\emptyset = \frac{k\lambda}{D} = \frac{8.16}{10} = 54.7\ degrees$$

Listing 1. Dieter Fox algorithm

```
1. Inputs: Distance ut,
2. sensor reading zt
3. sample set st ={(xt(i), wt(i)|i=1,...,n}
4. for i=1 to n do
5. xt = updateDist(xt,ut)
6. wt(i)=prob(zt|xt(i))
7. st+1=NULL
8. for i=1 to n do
9.  sample an index j from distribution given by the weights in St
10.  add (xt(j), wtj) to St+1
11. sample j to the set of new samples
12. return St+1
```

Listing 2. The proposed algorithm

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

```
1. Inputs: Distance ut,
2. sensor reading zt
3. sample set st={(sample1(i),sample2(i)|i=1,...,n}
4. for i=1 to n do
5. sample1 =updateDist(xt,ut)
6. wt(i)=prob(zt|sample1(i))
7. st+1=NULL
8. for i=1 to n do
9.  index j from distribution given by the weights in St
10.  add (sample1(j), sample2(j) to St+1
11. sample j to the set of new samples
12. return St+1
13.
```

Thus, the particles converge in a single location as the robot moves in the environment and senses different parts of the environment using a range sensor. The robot motion is detected using an odometer sensor. The particles in this process are updated:

1. Particles based on the change in the post and the detailed motion model has propagated.

2. The particles have assigned weights based on the likelihood of receiving the threshold of the sensor for each particle.

Then, based on these weights, a robot state estimation is extracted. The group of particles with the highest weight allow estimating the position of the robot.

Finally, the particles resample the adjusts particle positions and improve performance by changing the number of particles used. It is the main feature for adjusting to changes and keeping particles relevant for estimating the robot state.

The algorithm outputs the estimated position and covariance. These values allow estimating the mean and covariance of the highest weighted cluster of each particles set. For continuous tracking, it is possible to repeat these steps in a loop to propagate each particle, evaluate their likelihood, and obtain the best state estimate.

When working with a localization algorithm, the goal is to estimate the state of the system. For robotics applications, this determined state is usually a current position. The Monte Carlo Localization algorithm can specify this position as a three-element vector. The position corresponds to an x-y position [x, y], and angular orientation (z), theta. See Figure 2.

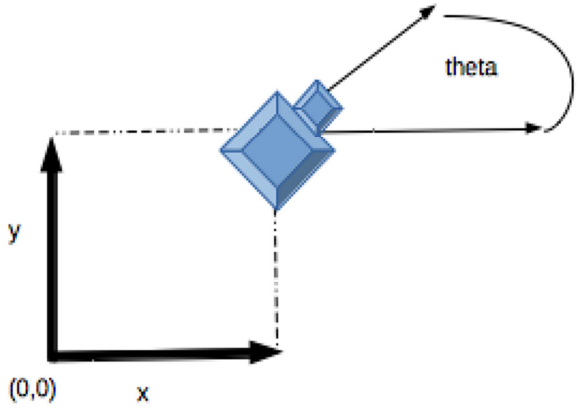

Fig. 2. Final position estimation

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

Particles are distributed around an initial position or sampled uniformly using global localization. The position is computed as the mean of the highest weighted cluster of particles once these particles have been corrected based on measurements.

Figure 3(a) shows the highest weighted group and the final robot position displayed over the samples particles. With more iterations of the algorithm and frequency corrections, the particles should converge to the real location of the robot. However, it is possible that particle sets can have high weights for false estimates and converge on the wrong location. If the unfortunate convergence occurs, it is necessary to resample the particles by resetting the algorithm with an updated initial position. See Figure 3(b).

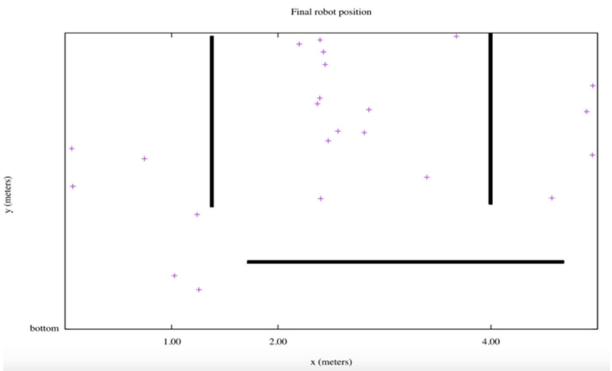

Fig. 3(a). Path trajectory and the final robot position

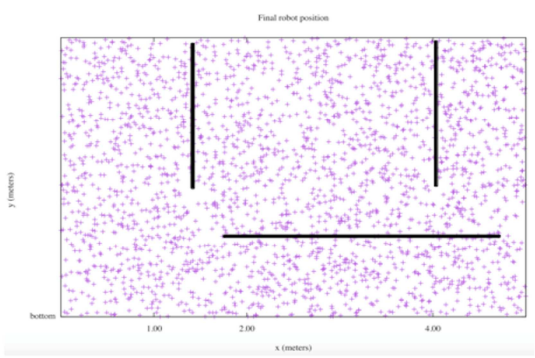

Fig. 3(b). Updated initial position

When Monte Carlo Localization algorithm is modified, it is necessary to specify the minimum and maximum particle range. A higher number of particles in the system, increase the likelihood that the particles converge on the real location. However, a lower particle number is faster but less precise.

The best number of particles adjusts dynamically within the limits based on the weights of particle sets. This adjustment allows reducing the number of particles over time so localization and the algorithm can run more efficiently.

3.1 PARTICLE DISTRIBUTION

Particles must be sampled across a specified distribution. To initialize particles in the state space, you can use either an initial position or the global localization.

The global localization, distributes particles across the expected state space. Global localization requires a larger number of particles to sample particles in the whole state and increase the likelihood; allowing convergence on the real state. This extensive distribution significantly reduces initial performance until particles begin to converge, and number of particles are reduced.

Global localization is set to false (0). Without global localization, you must specify the Initial position and initial covariance, (such as our case) which helps to localize the particles. When this original location it is used, particles are tightly grouped in an estimated state. A close particles set uses fewer of them and increases the speed and accuracy of tracking during the first iterations.

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

3.2 MOTION AND SENSOR MODEL

By default, The Monte Carlo Localization uses an ultrasonic sensor. This sensor contains specific parameters to calculate the range used; the response is in 2D and with this information, an environment map is created.

The sensor uses the parameters with full range measurements to calculate the likelihood of the measures using the current position of the robot. Without these parameters, some measurement errors can skew the state estimate or increase weight on foreign particles.

The sensor properties are:
1. Sensor position – The position of the range sensor about the robot location.
2. Sensor limits – The minimum and maximum range limits, measurement outside of these ranges in the likelihood calculation are not considered.
3. Number beams – Number of beams used to calculate the probability.
4. Frequency noise – Standard deviation of measurement noise.
5. Random size weight – Weight for a probability of random measurement. Set a small chance for random sizes. The default is 0.05.
6. Expected measurement weight – Weight for a probability of expected measurement.

The sensor also stores a map of the robot's environment as an occupancy grid. Set any unknown spaces on the map as free locations. Setting them to open locations prevents the algorithm from matching detected objects to these areas of the map.
The motion model for robot localization allows predicting**,** how particles should evolve throughout the time when resampling. It is a good representation of a robot kinematics. The motion model included by default with the algorithm is an odometer-based differential drive motion design.

Without a motion design, predicting the next step is more difficult. It is necessary, to consider all errors produced by the wheel encoders and the other sensors used in odometer sensor; these mistakes define the particle distribution.

It is possible to specify the error expected based on the motion of the robot and is defined as four-element vector. These elements are determined by weights on the standard deviations:

1. Rotational error due to rotational motion**.**
2. Rotational error due to translational motion**.**
3. Translational error due to translational motion**.**
4. Translational error due to rotational motion**.**

In robots with differential drive, when a robot moves from a starting position to a final position, the change in position is treated as:

1. Rotation to the last position**.**
2. Translation in a direct line to the last position**.**
3. Rotation to the goal orientation**.**

The algorithm proposed is shown in Figure 4. The Matlab source code is shown in the Annex 1.

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

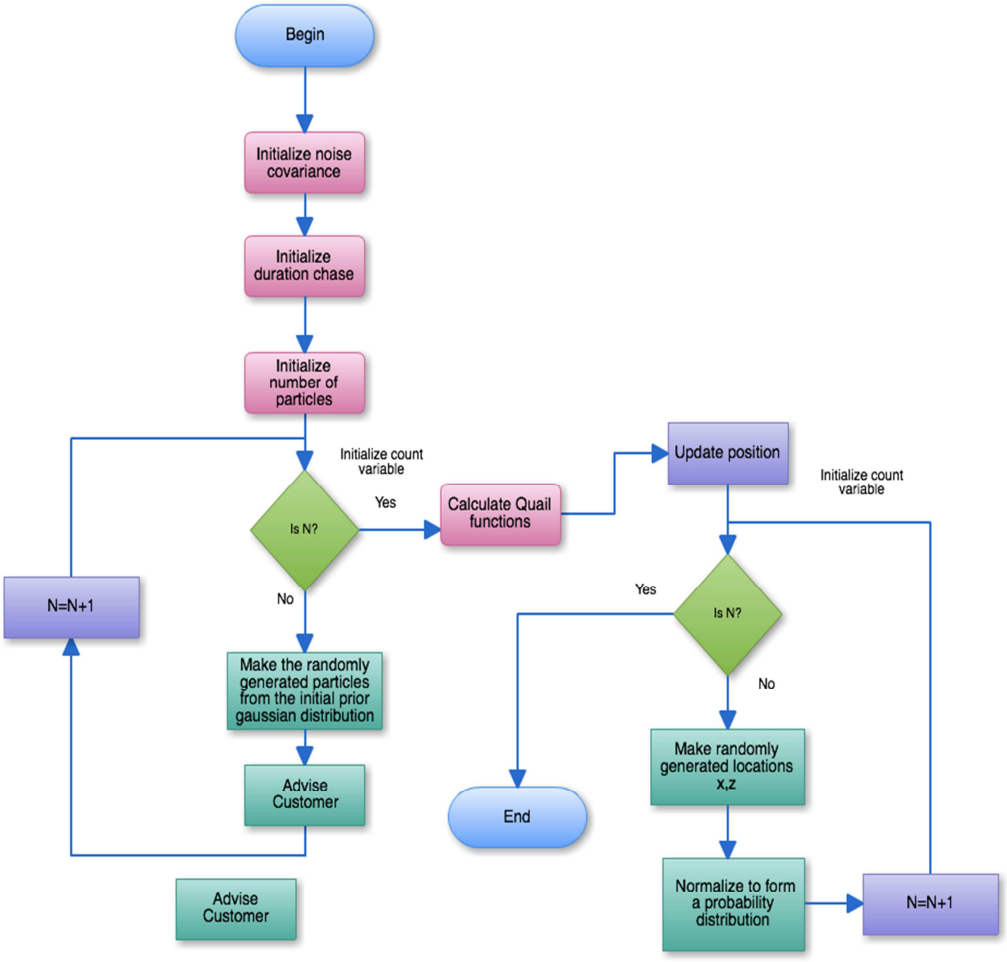

Fig. 4. Flow diagram of the proposed algorithm

3.3 Arduino implementation

The Arduino implementation is based on the source code of domain public where the 5V pin of the SEN136B5B is connected to the 5V pin on the board, the GND pin is connected to the GND pin, and the SIG (signal) pin is connected to digital pin 7 on the board. See Figure 5.

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

RESEARCH ARTICLE

Jesús Antonio Álvarez Cedillo, Elizabeth Acosta Gonzaga, Juan Carlos Herrera Lozada, Teodoro Álvarez Sánchez, Jacobo Sandoval Gutiérrez, Mario Aguilar Fernández

MECHANICS
Applied robotics

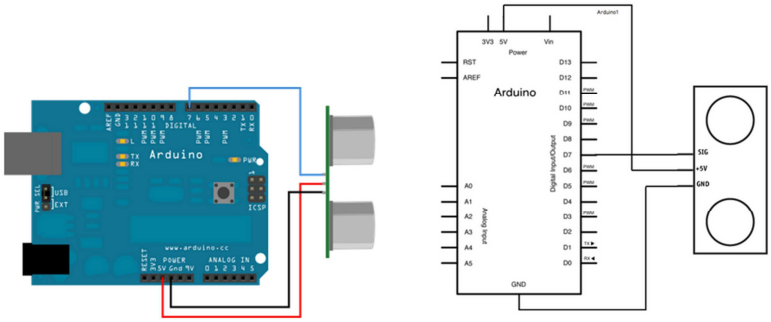

Fig. 5. Ping sensor connection

The basic implementation to read the distance using Ping sensors is shown in Listing 3.

Listing 3. The proposed Arduino code with the Z_t and u_t variables

```
1.
2.
3.
4.  const int pingPin = 7;
5.
6.  void setup() {
7.    // initialize serial communication:
8.    Serial.begin(9600);
9.    Zt = pulseIn(pingPin, HIGH);
10.
11. }
12.
13. void loop() {
14.     long Zt,  cm;
15.
16.   pinMode(pingPin, OUTPUT);
17.   digitalWrite(pingPin, LOW);
18.   delayMicroseconds(2);
19.   digitalWrite(pingPin, HIGH);
20.   delayMicroseconds(5);
21.   digitalWrite(pingPin, LOW);
22.
23.   pinMode(pingPin, INPUT);
24.   sample1 = pulseIn(pingPin, HIGH);
25.
26.   // convert the time into a distance
27.     ut = microsecondsToCentimeters(duration);
28.
29.   Serial.print(cm);
30.   Serial.print("cm");
31.   Serial.println();
32.
33.   delay(100);
34. }
35.
36. long microsecondsToCentimeters(long microseconds) {
37.   return microseconds / 29 / 2;
38. }
```


Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

4.- RESULTS AND DISCUSSION

4.1 THE PROPOSED MONTE CARLO METHOD

The results show the distribution of the particles around this initial value, see Figure 6(a). The calculation of the initial position used the following initial values: initial actual state 0.1; noise covariance in the system 1; duration the chase 75; number of particles 10. Figure 6(b) shows the mobile robot's positions, based on time and number of particles.

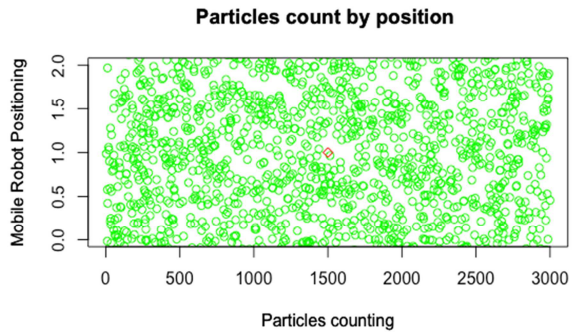

Fig. 6(a). Mobile robot's position around of this initial value of x, where the actual initial state is 0.1; noise covariance in the system =1; noise covariance in the measurement; duration the chase=75 with 10 particles and variance estimate = 2.

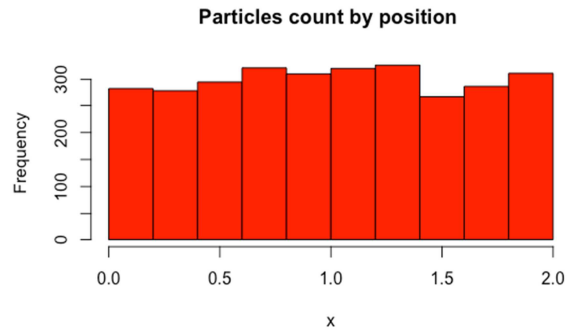

Fig. 6(b). Position estimation using count of particles where the actual initial state is 0.1; noise covariance in the system =1; noise covariance in the measurement; duration the chase=75 with 10 particles and variance estimate = 2.

Also, it is necessary to test the correct operation of the proposed algorithm. At this point was used the RoboSim software version 2.0.0. It is Java based portable simulator to visualize and understand the robot localization, path planning, path smoothing and PID controller concepts. It very flexible uses the Java language, is easy to use and it supports multiple platforms.

The comparison between the different steps of the robot with the RoboSim software was programmed. The software allows configuring two main features:
1. The filter settings
2. The runtime settings

The simulation used the values of Table 1.

Table 1. RoboSim Settings

Filter settings	Value	Runtime settings	Value
Sense noise	5	Turning angle	3
Robot Size	30	New Particles %	0

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

Particles	3000	Laser Range	200
Steering Noise	1	speed	20
Ghost Size	15	Unsampled %	0
Land Mark size	15	Laser angle	180
Forward noise	1.05	Bounded vision	on
Particle size	4	Show Ghost	on

The next step is run the simulation, for to do this, a modified script developed in Java language for the proposed algorithm was used. The program shows information for each position and with this, it is possible to know the path and each location. All the robot post is saved into a CSV File. Figure 7 shows the RoboSim simulation with the initial position.

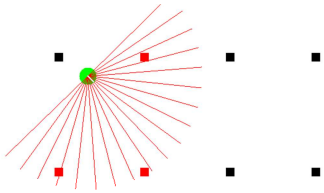

Fig. 7. Initial Position, the initial actual state is 0.1; Steering noise covariance in the system =1; Sense noise=5, Ghost size=15 covariance in the measurement; duration the chase=75 with 3000 particles and variance estimate = 2 with a particle size=4.

The position of the robot and the path was plotted in the cartesian plane using the R software. The positions with values in *x* and *y* axis are loaded in R software, they are shown in Table 2.

Table 2. Position estimated

X	Y
-32.38700344	-57.65217815
117.6129966	-57.65217815
-19.55949877	-72.9967252
130.4405012	-72.9967252
-7.552645292	-88.99158279
142.4473547	-88.99158279
3.600647043	-105.5929102
153.600647	-105.5929102
13.86980784	-122.7552042
23.22669003	-140.4314242

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

31.64564706	-158.573121
39.10360316	-177.1305694
28.36130774	80.68483087
178.3613077	80.68483087
19.56078268	62.72513269
169.5607827	62.72513269
9.832380453	45.25063155
159.8323805	45.25063155
-0.797234064	28.30922384
149.2027659	28.30922384

Therefore, the positions obtained are plotted on a Cartesian plane.

This analysis is targeted to developers of robotic applications in which position is critical, and application where accuracy matters. This should be more exploited in industrial applications.

The proposed algorithm shows how it is possible to achieve localization and positioning accuracy of a few millimeters at reference locations. This analysis is targeted to developers of robotic applications in which position is critical, and application where accuracy matters. This should be more exploited in industrial applications.

4.2 ARDUINO METHOD

It was used the Arduino board to store the information corresponding to sample 1, sample 2, the registered position, KLM and with this, we estimated the real position.

A small error of the position of 0.001 mm is observed when plotting a complete trajectory of 100 points in straight line, also, it was observed a deviation of 0.15 degrees. Table 1 shows the variables and the real position respect to estimated position.

Table 3. Real position versus estimated position

i	Real Position	KLM	Estimated Position
1	0	0	0
2	135	135.135	135.0675
3	120	120.12	120.06
4	105	105.105	105.0525
5	90	90.09	90.045
6	75	75.075	75.0375
7	60	60.06	60.03
8	45	45.045	45.0225
9	40	40.04	40.02

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

Therefore, it is necessary to calculate the variation in both methods, the position obtained is a little greater than the real value. This estimation uses the expression 1 to know the variations.

$$Epos = Loc - Rpos \qquad (1)$$

Where Epos is the estimated position, Loc is the measured position obtained by every method, and the Rpos is the real position. The estimation of the posts is shown in Figure 8.

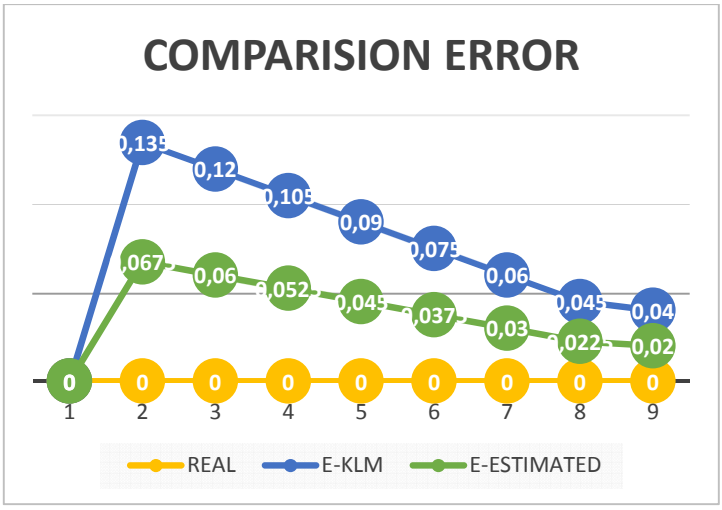

Fig. 8. A comparison based on the difference between the real position with the KLM and the proposed algorithm

The REAL (orange line) represents non error. The E-KLM (blue line) represents an estimated error using KLM from 0.04 up to 0.135. E-ESTIMATED (green line) has a minor error from 0.02 up to 0.067.

The proposed method shows a better performance, it is due to particle number used to produce the final position. When the Monte Carlo method is used, the sample number is the main parameter since it increases the accuracy and the execution time without increasing the memory size in the hardware.

5.- CONCLUSIONS

In this paper, we presented an analysis that thoroughly evaluated the accuracy of mobile robot localization and positioning system using a particle system.

This work, highlighted a new algorithm based on particles localization. The analysis is targeted to developers of robotic applications in which position is critical, and application where accuracy matters. This is more exploited in industrial applications.

The proposed algorithm shows how it is possible to achieve a localization and positioning accuracy of a few millimeters at reference locations The error computed in each case of the proposed algorithm was better than KLM algorithm. The results are consistent with those presented in [32]. These authors made a comparison of estimation of several algorithms for mobile robot navigation using Kalman Filters variants. The performance of the algorithm confirms that the measures presented in [34] are similar, where the particle number has a better precision (e.g. 1000 particles) and pose estimation than that produced by Kalman filtering techniques. However, the computational cost is greater.

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

This paper has proposed an algorithm to be applied in small embedded structures such as arm architectures, general applications based on Arduino, and numerical architectures on FPGAs or many-core systems. These systems are suitable for the proposed algorithm since they have low energy consumption, small memory and reduced processor capacity.

It is not limited to other robotic platform such as wheeled, aerial and multi-pot robots due an easy high-level programming.

BIBLIOGRAPHY

[1] Diop, M. et al. A Computer Vision-Aided Motion Sensing Algorithm for Mobile Robot's Indoor Navigation. 14th IEEE International Workshop on Advanced Motion Control (AMC). Auckland, New Zealand, IEEE. 400-405, 2016.

[2] Pandey, A. & Parhi, D. R. Autonomous mobile robot navigation in cluttered environment using hybrid Takagi-Sugeno fuzzy model and simulated annealing algorithm controller. World Journal of Engineering. Volumen: 13, Nom: 5. Pages: 431-440, 2016.

[3] Pandey, A. & Parhi, D. New algorithm for behaviour-based mobile robot navigation in cluttered environment using neural network architecture. World Journal of Engineering. Volumen: 13, Nom: 2, Pages: 129-141, 2016a.

[4] Wen, S. et al. Improved Artificial Bee Colony Algorithm Based Optimal Navigation Path for Mobile Robot. 12th World Congress on Intelligent Control and Automation (WCICA). Guilin, Peoples R China. Jun 12-15, 2016. IEEE: 2928-2933, 2016.

[5] Miloud, H. & Abdelouahab, H. Improving mobile robot navigation by combining fuzzy reasoning and virtual obstacle algorithm. Journal Of Intelligent & Fuzzy Systems. Volumen: 30, Nom: 3. Pages: 1499-1509, 2016.

[6] Mohanty, P. K., Kumar, S. & Parhi, D. R. A New Ecologically Inspired Algorithm for Mobile Robot Navigation. Conference: 3rd International Conference on Frontiers in Intelligent Computing - Theory and Applications (FICTA), Bhubaneswar, India, Volumen: 327. Pages: 755-762, 2015.

[7] Jiang, J. et al. Cognitive response navigation algorithm for mobile robots using biological antennas. Robotica. Volumen: 32, Nom: 5. Pages: 743-756, 2014.

[8] Mohanty, P. K. & Parhi, D. R. A New Real Time Path Planning For Mobile Robot Navigation Using Invasive Weed Optimization Algorithm. Conferencia: ASME Gas Turbine India Conference Ubicación: New Delhi, India, V001T07A002, 2014.

[9] Huang, Hsu-Chi, Wu, Ter-Feng & Huang, Chun-Yu. A Metaheuristic Artificial Immune System Algorithm for Mobile Robot Navigation. Conferencia: 10th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP). Waseda Univ, Kitakyushu, Japan, Pages: 803-806, 2014.

[10] Pang, M. et al. MGRO Recognition Algorithm-Based Artificial Potential Field for Mobile Robot Navigation. Journal of Sensors. Volume 2016, Article ID 1959160, 7 pages, 2016.

[11] Panati, S. ; Baasandorj, B. & Chong, K. T. Harmonic Potential Function Based Algorithm for Autonomous Mobile Robot Navigation. Advanced Science Letters. Volumen: 21, Nom: 12, Pages: 3662-3666, 2015.

[12] Mohanty, P. & Parhi, D. A new hybrid optimization algorithm for multiple mobile robots navigation based on the CS-ANFIS approach. Memetic Computing. Volumen: 7, Nom: 4. Pages: 255-273, 2015.

[13] Abadi, M. H. B., Oskoei, M. A. & Fakharian, A. Mobile Robot Navigation using Sonar Vision Algorithm applied to Omnidirectional Vision. 2015 AI & Robotics IRANOPEN. Qazvin, IEEE, 2015.

[14] Jalel, S., Marthon, P. & Hamouda, A. Optimized NURBS Curves Modelling Using Genetic Algorithm for Mobile Robot Navigation. Computer Analysis of Images and Patterns, CAIP 2015, PT I., Lecture Notes in Computer Science. Volumen: 9256. Pages: 534-545, 2015.

Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

[15] Ismail, A. R. et al. Implementation of Cognitive Mapping Algorithm for Mobile Robot Navigation System. Conference: 4th World Congress on Information and Communication Technologies (WICT) Melaka, Malaysia, Pages: 280-284, 2014.

[16] Jan, G., et al. A Computationally Efficient Complete Area Coverage Algorithm for Intelligent Mobile Robot Navigation. Conference: International Joint Conference on Neural Networks (IJCNN) Beijing, Peoples R China, Pages: 961-966, 2014.

[17] Dang Quoc Khanh; Suh, Young-Soo. Mobile Robot Destination Generation by Tracking a Remote Controller Using a Vision-aided Inertial Navigation Algorithm. Journal of Electrical Engineering & Technology. Volumen: 8, Nom: 3. Pages: 613-620, 2013.

[18] J. S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 1998.

[19] P. Jensfelt, D. Austin, O. Wijk, and M. Andersonn. Feature based condensation for mobile robot localization. In Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 2000.

[20] M. Quigley, D. Stavens, A. Coates, and S. Thrun. Sub-meter indoor localization in unmodified environments with inexpensive sensors. In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2010.

[21] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo Localization for Mobile Robots. Artificial Intelligence, 128, 2001.

[22] F. Duvallet and A. Tews. Wifi position estimation in industrial environments using Gaussian processes. In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2008.

[23] Ali Azarbayejani and Alex Pentland. "Recursive Estimation of Motion, Structure, and Focal Length," IEEE Trans. Pattern Analysis and Machine Intelligence, 1995, 17(6).

[24] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Image-based Monte-Carlo localization with omnidirectional images. Robotics & Autonomous Systems, 48(1):17–30, 2004.

[25] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), 1999.

[26] P. Elinas and J.J. Little. MCL: Monte-Carlo localization for mobile robots with stereo vision. In Proc. of Robotics: Science and Systems(RSS), 2005.

[27] Borenstein, J., Everett, B., and Feng, L. Navigating Mobile Robots: Systems and Techniques. A. K. Peters, Ltd., 1996.

[28] Burgard, W., Fox, D.; Hennig, D.; and Schmidt, T. Estimating the absolute position of a mobile robot using position probability grids. Proc. of AAAI-96, 1996.

[29] Burgard, W., Derr, A., Fox, D., and Cremers, A. Integrating global position estimation and position tracking for mobile robots: The Dynamic Markov Localization approach. Proc. Of IROS-98, 1998b.

[30] Dieter Fox, W. Burgard, F. Dellaert, S. Thrun; Robust Monte Carlo Localization: Efficient Position Estimation for Mobile Robots, Proceedings of the National Conference on Artificial Intelligence, 343-349, JOHN WILEY & SONS LTD, 1999.

[31] S. Thrun, W. Burgard, D. Fox.; Probabilistic Robotics. MIT Press, 2005.

[32] Guney, S.& Bilen, M., The Comparison of Estimation Algorithms for Mobile Robot Navigation. 24th Signal Processing and Communication Application Conference (SIU). Zonguldak, Turkey. May 16-19, 2016. IEEE: 797-800, 2016.

[33] Nirmala, G., Geetha, S. & Selvakumar, S., Intellectual Method of guiding Mobile Robot Navigation using Reinforcement Learning Algorithm. IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, India, 2015.

[34] Zongwen Xue and Howard Schwartz, A Comparison of Mobile Robot Pose Estimation using Non-linear Filters: Simulation and Experimental Results, International Journal of Mechatronics and Automation (IJMA), Vol. 5, No. 2/3, 2015

[35] Chavez-Estrada, F., Sandoval-Gutierrez, J., Herrera-Lozada, J., Alvarez-Cedillo, J., Olguin-Carbajal, M.. (2017). MICRO GENETIC ALGORITHM BASED ON A NOVEL ESTIMATOR FOR THE QUADRUPED ROBOT LOCOMOTION. DYNA, 92(2). 136. DOI: http://dx.doi.org/10.6036/8215.

SUPPLEMENTARY MATERIAL

```
clear all
close all
clc

set(0,'DefaultFigureWindowStyle','docked')
x = 0.1;
x_N = 1;
x_R = 1;
T = 75;
N = 10;.
V = 2;
x_P = [];
for i = 1:N
    x_P(i) = x + sqrt(V) * randn;
end
figure(1)
clf
subplot(121)
plot(1,x_P,'.k','markersize',5)
xlabel('time step')
ylabel('flight position')
subplot(122)
hist(x_P,100)
xlabel('flight position')
ylabel('count')
pause
z_out = [x^2 / 20 + sqrt(x_R) * randn];
x_out = [x];
x_est = [x];
x_est_out = [x_est];
for t = 1:T
x = 0.5*x + 25*x/(1 + x^2) + 8*cos(1.2*(t-1)) +  sqrt(x_N)*randn;
    z = x^2/20 + sqrt(x_R)*randn;
for i = 1:N
       x_P_update(i) = 0.5*x_P(i) + 25*x_P(i)/(1 + x_P(i)^2) + 8*cos(1.2*(t-1)) + sqrt(x_N)*randn;
z_update(i) = x_P_update(i)^2/20;
P_w(i) = (1/sqrt(2*pi*x_R)) * exp(-(z - z_update(i))^2/(2*x_R));
    end
   P_w = P_w./sum(P_w);
   figure(1)
    clf
    subplot(121)
 plot(P_w,z_update,'.k','markersize',5)
    hold on
    plot(0,z,'.r','markersize',50)
    xlabel('weight magnitude')
    ylabel('observed values (z update)')
    subplot(122)
plot(P_w,x_P_update,'.k','markersize',5)
 hold on
    plot(0,x,'.r','markersize',50)
```


Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com

Jesús Antonio Álvarez Cedillo, Elizabeth Acosta Gonzaga, Juan Carlos
Herrera Lozada, Teodoro Álvarez Sánchez,
Jacobo Sandoval Gutiérrez, Mario Aguilar Fernández

```
    xlabel('weight magnitude')
    ylabel('updated particle positions')
    pause

figure(1)
    clf
    subplot(131)
plot(0,x_P_update,'.k','markersize',5)
    title('raw estimates')
    xlabel('fixed time point')
    ylabel('estimated particles for flight position')
    subplot(132)
plot(P_w,x_P_update,'.k','markersize',5)
    hold on
    plot(0,x,'.r','markersize',40)
    xlabel('weight magnitude')
    title('weighted estimates')

for i = 1 : N
        x_P(i) = x_P_update(find(rand <= cumsum(P_w),1));
    end


x_est = mean(x_P);
subplot(133)
plot(0,x_P_update,'.k','markersize',5)
    hold on
    plot(0,x_P,'.r','markersize',5)
    plot(0,x_est,'.g','markersize',40)
    xlabel('fixed time point')
    title('weight based resampling')
    pause
x_out = [x_out x];
    z_out = [z_out z];
    x_est_out = [x_est_out x_est];
    end
t = 0:T;
figure(1);
clf
plot(t, x_out, '.-b', t, x_est_out, '-.r','linewidth',3);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('time step'); ylabel('Quail flight position');
legend('True flight position', 'Particle filter estimate');
```


Publicaciones DYNA SL -- c) Mazarredo nº69 - 4º -- 48009-BILBAO (SPAIN)
Tel +34 944 237 566 – www.dyna-newtech.com - email: info@dyna-newtech.com