Fast Position Estimation for Autonomous Mobile Robot Navigation

ALVARO SANCHEZ-MIRALLES, MIGUEL ANGEL SANZ-BOBI
Instituto de Investigación Tecnológica
Universidad Pontificia Comillas
Alberto Aguilera, 23. 28015 Madrid
SPAIN

Abstract: - One of the problems in mobile robotics is the estimation of the robot position in the environment. In this paper we propose a model, called positioning model, for estimating a confidence interval of the robot position, in order to compare it with the estimation made by a dead-reckoning system. Both estimations are fused with heuristic rules. The positioning model is useful to estimate the robot position with or without previous knowledge of the previous position. Furthermore, it is possible to define the degree of previous knowledge of the robot position, allowing to make the estimation adaptive by varying this knowledge degree. This model is based on a one-pass neural network which could adapt itself in real time conditions and could learn the relationship between exteroceptive sensors measurements and the robot position.

Key-Words: - First location problem, RTDENN, neural network, continuous mobile robot relocalization.

1 Introduction

An autonomous mobile robot must be able to know the environment before executing some of the tasks for which it was designed. Executions mobile robot movements as well as the success in performing its tasks depend on the quality of the measurements taken by its sensors. An autonomous mobile robot has generally an odometer, which allows it to obtain information about its position. However, the measurements taken by the sensors are imprecise and with many limitations.

The main source of error comes from the estimation of the robot position. The imprecise measurement that the robot odometer gives, gets worse by the accumulation of errors due to the sliding of the wheels or to its own limitations. For that reason, it is necessary a method for estimating robot position using information of the environment.

Position estimation has been approached in different and clearly differentiated ways:

- *The estimation based on the landmarks detection [1].*
- *The estimation based on matching exteroceptive measurements with a metric map of the environment [2][3][6].*

A new technique for estimating robot position of autonomous robots, based on the positioning model, (PM) is described in this article. The mentioned model learns the relationship between the measurements of exteroceptive sensors and the robot position. The PM doesn't distinguish between first location problem and robot relocalization, because it approaches both problems in the same way. Furthermore, it is possible to define the degree of previous knowledge of the robot position. This allows to make estimation adaptive, by varying this knowledge degree. As a result the model is more robust against failures due, for example, to a bad correction of the position, sliding of wheels, etc.

The estimation of the position is not only reduced to give a value, but also to calculate a set of confidence intervals within which the position and orientation of the robot should be found at a 99% confidence. The model is adapted in real time to changes in the environment.

2 Positioning model

The PM estimates the robot position and orientation as it moves. As a result, confidence intervals for each of the variables to be estimated are provided, within which

robot position and orientation must be at a 99% confidence.

There are two types of estimation in real applications:

- The estimation without previous knowledge of the robot position (first location problem), called in this paper *robot localization*.
- The estimation with knowledge of the robot position at the previous moment (relocalization problem).

In this section it is proposed a model which doesn't difference the type of estimation; it means that the model to estimate the robot position with or without previous knowledge is unique. To achieve this, positioning model has a parameter that measures the knowledge of the previous robot position, called *positioning excitation threshold* (*PET*), see section 2.5. Thanks to this parameter the concept of two states disappears: knowledge and ignorance of the robot position at the previous moment. On the other hand, a continuum of states from knowledge to complete ignorance through different degrees of knowledge of previous robot position, appear.

If it is possible to vary gradually this parameter and use it as an input to the position estimation algorithm, then it is possible to control it according to the estimation of the resulting confidence intervals. This causes a feedback as shows fig.1, where a module that determines the strategy to follow by the position estimation algorithm is involved.

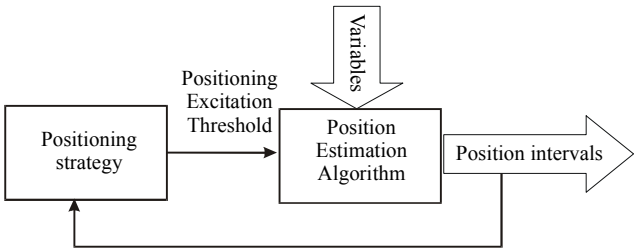

Fig. 1. Positioning model

The position module works as follows:

- If it is the first time it starts working and the robot does not know its position, then the *PET* becomes 0, with which it would be able to apply the localization algorithm. Once we know certainly where the robot is, then the threshold is increased to a level of 1/3, because it is the maximum and it corresponds to the minimum excitation of a neuron (position estimation algorithm uses a neural network which is explained in section 2.2) with which the confidence intervals of 99% confidence are obtained.
- If we know approximately where the robot is, then the *PET* is established at 1/3. If we do not find active neurons, either the region is not known for the PM, or the estimation of the robot position is not the correct one. In order to try to correct this last point, the *positioning threshold* is diminished by reducing the value. If the region is not known or the robot gets lost, it is necessary to reduce the threshold to 0.
- If the area where the robot moves is unknown, the application of this algorithm does not have sense. This is detected because the robot loses itself all the time and the threshold is 0.
- It may occur that the localization of the lost robot will not be successful giving an erroneous solution. This is not a problem, since in a short time the robot will get lost, because the PM is not coherent with the sensor measurements.

In the following subsections the PM is explained in detail. It is based on a neural network called RTDENN, explained in section 2.2, and on the position estimation algorithm explained in section 2.5. In order to explain this algorithm, it is first described the localization algorithm of the robot, which is a specific case of first one and helps to understand it, see section 2.4.

2.1 Description

The positioning model (PM) is based on a neural network, called positioning neural network (PNN), which learns the relationship between exteroceptive sensorial measurements and the robot position

The PNN is very useful to estimate with uncertainty the robot position in the environment from measurements taken by exteroceptive sensors. It has as inputs the position and orientation of the robot, in addition to the exteroceptive sensor measurements. The output of the network is an estimation of the probability density function of the input variables, see fig. 2.

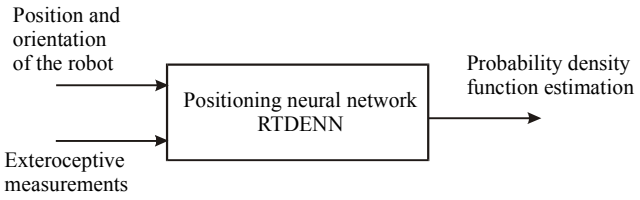

Fig. 2: Positioning Neural Network

The implantation is a key point, since the model must fulfil certain requirements that make it suitable for mobile robots. The requirements are listed below:

- To work in real time.
- A compromise between the use of memory and learning precision.
- The model must be easy to interpret so that its information should be also easy to use by the algorithms, which estimate the robot position.
- Facility to adapt to changing situations.
- Possibility of learning without any previous knowledge.

The use of the RTDENN which fulfils the above mentioned requirements is proposed. The RTDENN is adapted in real time, allowing to optimize the use of memory according to the required precision. One of the main advantages is that it can learn starting from no knowledge in real time conditions, which allows the robot to move in totally unknown environments. In spite of being multi-dimensional, this network is easy to interpret as you can interpret the RTDENN structure. That is, neurons can be projected in different dimensions, the centers of which represent the robot working patterns and which covariance matrices show the area size where each pattern works.

2.2 RTDENN

This section aim is to describe the RTDENN (real time dynamic ellipsoidal neural network).

2.2.1 Description

Let R be the M-dimensional workspace where the robot has to work. Each dimension of R corresponds to a variable collected by the robot. R can be divided in sub-workspaces or regions $R_i \subset R$.

Let $\mathbf{X}_N = \{\mathbf{x}_i\}_{i=1}^{N}$ be a set of sample vectors containing values of variables monitored by the mobile robot. Each $\mathbf{x}_i = \{x_{ij}\}_{j=1}^{M}$ belongs to $R \subset \Re^M$. The RTDENN consists of a set of neurons $NN_I = \{nn_i\}_{i=1}^{I}$, see Fig. 3, where each neuron nn_i is specialized in a particular sub-workspace R_i. Every set of sample vectors included in every sub-workspace R_i represented by the neuron nn_i can be characterised by the following RTDENN parameters:

- The number of sample vectors that belongs to the region or sub-workspace, K_i.
- One vector $\boldsymbol{\mu}_{K_i}^{i}$ obtained by the estimation of the mean values of the sample vectors of that region. $\boldsymbol{\mu}_{K_i}^{i} = \sum_{j=1}^{K_i} \frac{\mathbf{x}_j^i}{K_i}$.
- A MxM covariance matrix of the sample vectors of a sub-workspace noted as $\boldsymbol{\Omega}_{K_i}^{i}$ with elements $v_{(n,m)\cdot K_i}^{i} \mid n,m = 1 \cdots M$.

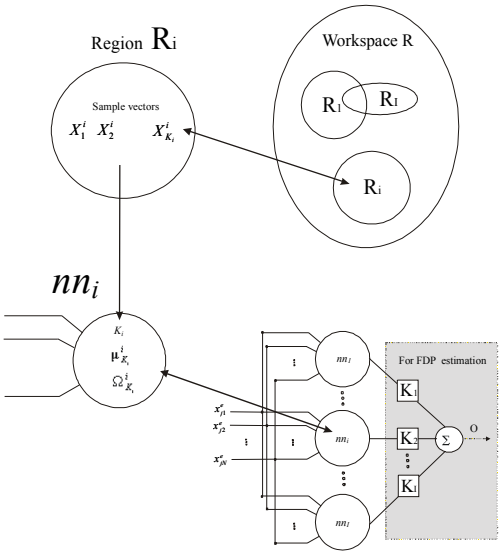

Fig. 3: RTDENN structure

2.2.2 RTDENN neuron excitation

Neurons are excited by a set of examples $X_H^e = \{x_i^e\}_{i=1}^{H}$. These examples are characterized with the same parameters as neurons are; that is, the number of examples of the input set H, the mean of examples of the input set $\boldsymbol{\mu}_H^e$ and the covariance matrix $\boldsymbol{\Omega}_H^e$.

An i-esima neuron excitation degree is determined as follows:

$$exc_i = \sqrt{\frac{1}{(\boldsymbol{\mu}_H^e - \boldsymbol{\mu}_{K_i}^i)^T \cdot (\boldsymbol{\Omega}_{K_i}^i)^{-1} \cdot (\boldsymbol{\mu}_H^e - \boldsymbol{\mu}_{K_i}^i)}}$$

This excitation is characterized as follows:

- Maximum excitation exc_{max}: if $exc_i >= 1$. Above this excitation, there is 68% of examples, and in this case examples reinforce the neuron log-likelihood.
- Minimum excitation exc_{min}: if $1 > exc_i >= 1/3$. Below that excitation, there is 1% of examples, and so it is considered that those examples are out of the neuron scope.
- Without excitation: If $exc_i < 1/3$.

2.2.3 Network training

The training algorithm is a *one-pass learning* [4] that consists of several cycles in which the network is excited by different examples. Then activated neurons are selected (a neuron is active when its excitation is greater than the minimum excitation level, exc_{min}). If there isn't any activated neuron, it means that there is an area which is not modeled by the network and so, a neuron is added. On the other hand if there are activate neurons, their parameters are updated using the following recursive equations:

- Number of examples that represents a neuron:

$$K_i = \frac{x-H}{x} K_i + H \quad \text{(x is a forgotten factor [4])} \quad (1)$$

- Mean recursive equation

$$\boldsymbol{\mu}_k = \frac{k-H}{k} \cdot \frac{ff-H}{ff} \cdot \boldsymbol{\mu}_{k-n} + \frac{H}{k} \cdot \boldsymbol{\mu}_n \quad (2)$$

- Covariance recursive equation:

$$v_{(x,y)\cdot k} = \frac{k-H}{k} \cdot \frac{ff-H}{ff} \cdot (v_{(x,y)\cdot k-H} + \boldsymbol{\mu}_{x\cdot k-H} \cdot \boldsymbol{\mu}_{y\cdot k-H}) +$$

$$+ \frac{H}{k} \cdot (v_{(x,y)\cdot H} + \boldsymbol{\mu}_{x\cdot H} \cdot \boldsymbol{\mu}_{y\cdot H}) - \boldsymbol{\mu}_{x\cdot k} \cdot \boldsymbol{\mu}_{y\cdot k} \quad (3)$$

where H is the number of new examples taken by the robot during its movement, k represents K_i simplifying the notation and x is the number of samples that the neuron needs to remember beginning from the last one.

As the number of neurons increase, two neurons could be merged when they are very similar.

2.2.4 Merge of neurons

Two neurons are merged when they are specialized on very similar regions. The similarity degree is measured using a chi-square test applied to the Mahalanobis distance between them. The new parameters of the merged neuron are calculated using the recursive equations previously described.

2.3 Estimation of the most probable values of a variable

Once the PNN is trained, the value of an input variable can be estimated, knowing the value of all other input variables, called *bias variables*. The steps to follow are shown in fig. 4 (a two-dimensional case for better comprehension) and they are explained below:

1. To excite network with the values of input bias variables.
2. To detect neurons, which are activated if the network is projected over bias variables.
3. To calculate intersection points between non-projected neurons and the straight line parallel to the dimension of the variable to estimate and which crosses the bias point (given by the bias variables).
4. Once the two intersection points have been found (p^i_{max}, p^i_{max}) of each active neuron (NN_i), it is calculated the probability:

$$p^i = \left(\frac{\mathbf{p}^i_{min} + \mathbf{p}^i_{max}}{2} - \boldsymbol{\mu}^i_{K_i} \right)^T \cdot (\boldsymbol{\Omega}^i_{K_i})^{-1} \cdot \left(\frac{\mathbf{p}^i_{min} + \mathbf{p}^i_{max}}{2} - \boldsymbol{\mu}^i_{K_i} \right)$$

with which each one has been activated. The two values (p_{min}, p_{max}) determine the confidence intervals within which the estimated variable estimated should be (at 99% probability). They are calculated using the following equations:

$$p_{min} = \frac{\sum_i p^i \cdot \mathbf{p}^i_{min}}{\sum_i p^i} \qquad p_{max} = \frac{\sum_i p^i \cdot \mathbf{p}^i_{max}}{\sum_i p^i}$$

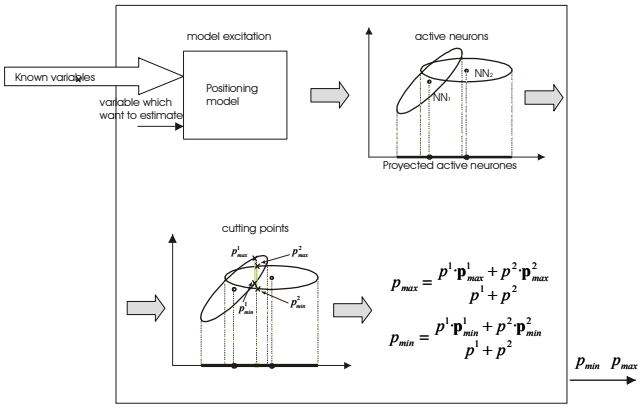

Fig. 3. Estimation of the most probable values of a variable

2.4 Localization algorithm

The localization algorithm estimates the robot position without previous knowledge of its previous position. This algorithm is a specific case of the position estimation algorithm, which is explained in 2.5.

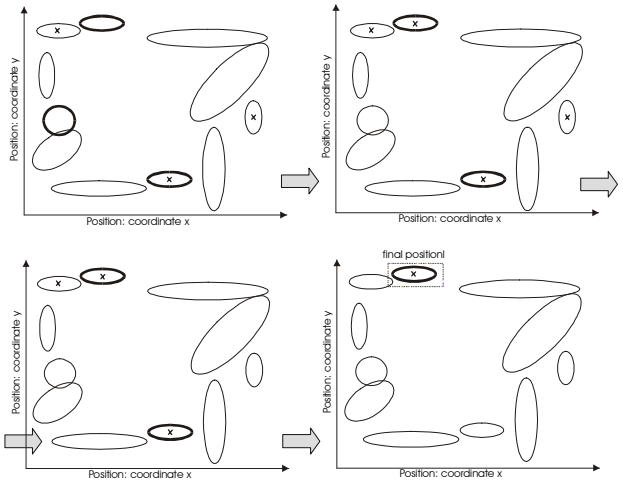

Fig. 4. Localization algorithm

The algorithm consists of several cycles in which, while the robot moves, some possible positions and orientations are selected. Each cycle takes into account both the information of the previous cycle and the sensor measurements. After each cycle positions and orientations become more restricted. Just when only one position is valid, the algorithm finished. The number of cycles is variable depending on the region where the robot is moving. Each cycle consists of several points which are explained below, see Fig. 5 (only PNN neurons projected on the x and y coordinates are shown):

1. The strategy of this point is to look for those environment positions in which it is coherent to have the measurements taken by the sensors.
2. Once coherent positions are found from a sensorial point of view, select those who remain unchanged as the robot moves, are selected. The parameter which decides what position is coherent is the *localization excitation threshold.*
3. If there is only one possible position, the robot has been localized and the confidence intervals of the position should be calculated as commented in section 2.3. Otherwise go back to the point 2.

The only parameter of the algorithm is the *localization excitation threshold.* If this threshold is reduced too much, then the algorithm convergence is quicker but it is more uncertain that the solution should be correct. If the threshold is increased, then the algorithm converges slower but the solution obtained is more robust.

2.5 Position estimation algorithm

This algorithm uses PNN to estimate the robot position. Its aim is to provide confidence intervals within which the robot position should be. It is calculated taking into account the sensor measurements and the robot position at the previous moment (imprecisely). In fig.6 is showed the inputs and outputs of the algorithm (two-dimensional case).

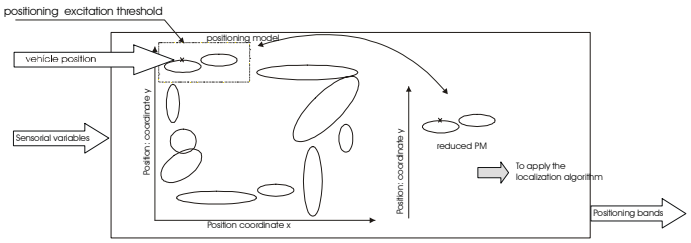

Fig. 5. Position estimation algorithm

Next it is described the steps the algorithm consists of:

1. In order to reduce the number of calculations, the positioning neural network is reduced to an equivalent network formed by those neurons representative of the environment in which the robot position and orientation are similar to those at the previous moment. The final network will have more or less neurons according to this similarity definition, which is determined by what is called p*ositioning estimation threshold (PET).* The model obtained in this way is called *reduced positioning neural network* (NNRP).
2. To apply the localization algorithm to the NNRP. As a result, confidence intervals are calculated.

As it can be shown, the position estimation algorithm is a generalization of the localization algorithm. Even more, if PET becomes 0, then both algorithms are equal.

2.6 Conflict resolution

As it was previously commented, PM estimates confidence intervals of the robot position. At the same time, a dead-reckoning system can give an estimation of the robot position and orientation as well. Both estimations must be compatible; that is to say, the position estimated by the dead-reckoning system should be within the confidence intervals estimated by the PM. If not, there is a conflict which is resolved using heuristic rules. This conflict could be due to:

- The position estimation is not correct due to dead-reckoning errors.
- The confidence interval estimation is not right due to the position estimation algorithm has found a robot workspace similar to another one in the environment.

3 Experimental results

Some experiments with the RTDENN have been done using the low cost mobile robot "UMI". It consists of 8 ultrasonic rangers (SRF04), 5 infrared rangers (GP2D12), one compass (VECTOR 2X) and an odometric system (one HP encoder in each wheel of the robot). Fig. 7 shows the layout of the robot.

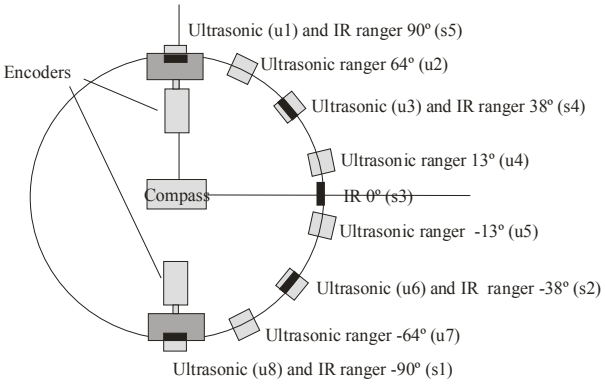

Fig. 7: Layout of the robot used in the experiments.

In order to difficult the robot navigation, the compass is deactivated. So the only way the robot has for estimate its position using propioceptive sensors is through dead-reckoning. In those conditions, it is necessary to apply the PM for achieving a good navigation.

All these experiments have been carried out in the environment shown in fig. 8.

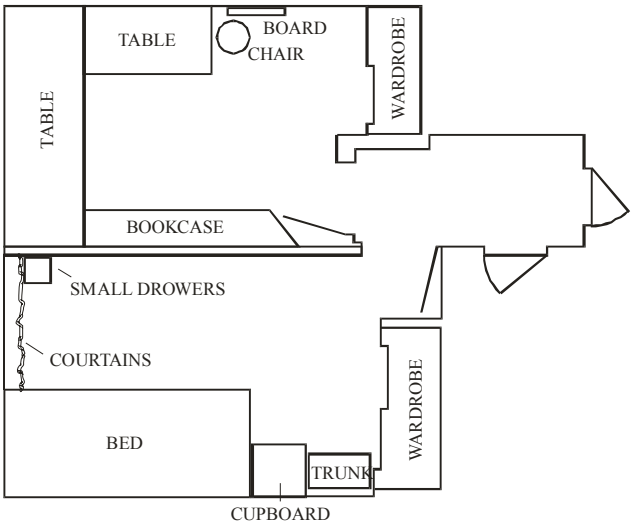

Fig. 8: Environment used in the experiments

The coordinate origin is situated in the corner formed by the bookcase and the table. So bookcase is the axis X and table is the axis Y. The robot position is measured in centimeters, and the orientation is measured in grades in a range of -180º to 180º.

In order to know where the robot is (for validation purposes), some landmarks were made on the floor by the robot as it moves. These landmarks are used to check the robot position through the experiments.

3.1 Case 1: Position estimation using PM

Next an experiment has been performed for estimating the robot position, knowing exactly the location of the robot at the starting point. This experiment consists of several cycles represented in fig. 9. It consist of three graphics corresponding, from up to bottom, the evolution of X coordinate, the evolution of Y coordinate and the evolution of robot orientation. Each cycle is represented by four points: upper limit (in red color) and lower limit (in blue color) of the estimated interval, the estimated position (in green color) and the real position (in cyan color). Additionally, it is represented with arrows the instants in which the robot has corrected its position.

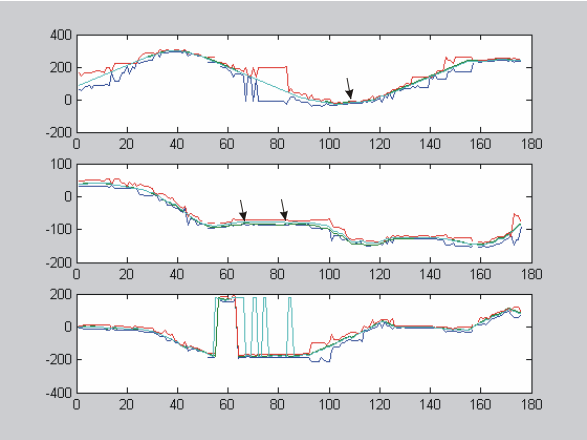

Fig. 9. Evolution of the X and Y coordinate of the robot position and the orientation.

Besides, in fig. 10 it is drawn the real (in green color) and estimated (in blue color) path followed by the robot in its navigation. The X and Y axis represent the X and Y coordinates of the robot position.

As it may be seen, robot can navigate in the environment thanks to PM.

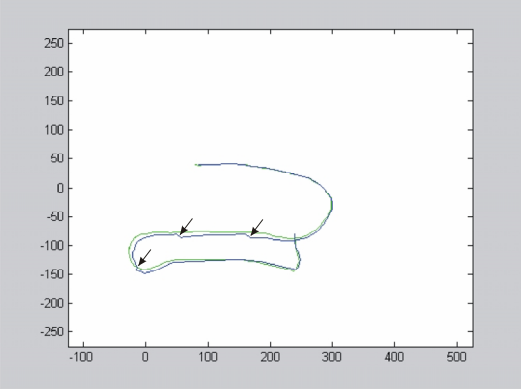

Fig. 10. Path followed by the robot

3.2 Case 2: First location problem using PM

Next an experiment has been performed with the aim of estimating the robot position when there isn't any knowledge of the starting location of the mobile robot. Fig. 11.a).b).c) shows, in yellow color, the PNN neurons projected on two dimensions, X and Y coordinates of the robot. A neuron is represented by its mean (the center of the ellipse) and by its covariance matrix (the ellipse itself indicates 1/3 of its influence region, that is to say, the region where the neuron could be activated) Neurons activated, because exteroceptive measurements match with the neuron influence region, are marked in red color. The position and the path followed by the robot are represented with a black cross and a black line respectively. At first, there are many activated neurons of the PNN (fig. 11.a). When the robot knows its position approximately, there is only one neuron activated (fig. 11.c).

On the other hand, fig. 11.d) shows the real (in green color) and estimated (in blue color) path followed by the robot when there is knowledge about its location.

Finally, in fig. 12 is represented the evolution of the robot position, in a way similar to fig. 9. At first, none estimated position intervals are given, because the robot doesn't know where it is.

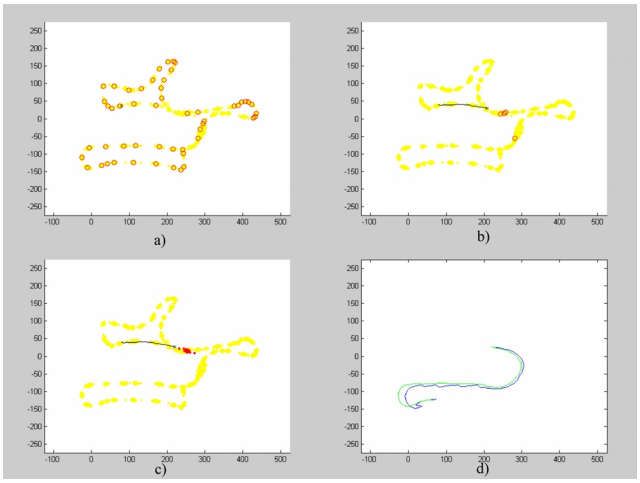

Fig. 10. a) b) c) The PMM: projected active neurons (in red color) in the 1st, 21st ,23rd cycle respectively. d) Path followed by the robot after locates itself.

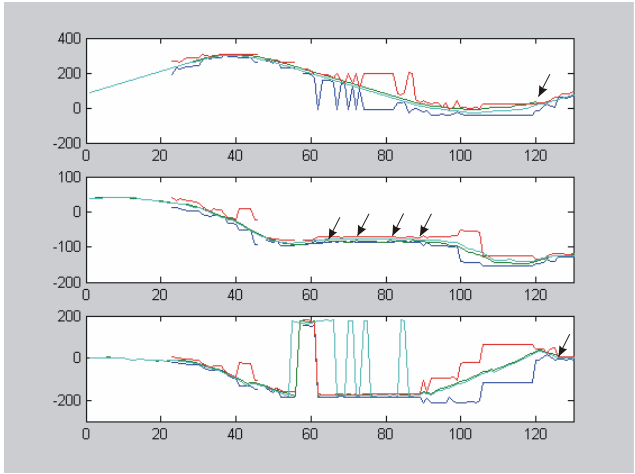

Fig. 11. Evolution of the X and Y coordinate of the robot position and the orientation.

4 Conclusion

A new method has been proposed for continuous robot positioning. It allows keeping a robot continuously localized by estimating its position using the estimation from a dead-reckoning system and comparing it with the estimation made by the PM. If they differ, a corrected estimation of the position is calculated heuristically.

A positioning model based on the positioning neural network has been developed. This model relates the sensor measurements with the robot position, so that given some sensor measurements, it is very simple to obtain an estimation of the robot position and vice versa.

The positioning model doesn't distinguish between first location problem and position estimation. This makes the method more robust, since it doesn't mind the robot was lost due to an erroneous correction of the robot position. Also the model could adapt itself in real time conditions, allowing simultaneous localization and model adaptation.

References:

[1] Robin R, Murphy. *Introduction to AI robotics*. 2000 MIT Press.

[2] J.M. Armingol. *Localización geométrica de robots autónomos.* PhD U Carlos III Madrid. 1997

[3] J. Boreenstein, B. Everett. *Navigating Mobile Robots: Systems and Techniques.* A. K. Peters, Ltd Wellesley, MA, 1996.

[4] Donald F. Specht. *A General Regresion Neural Network.* IEEE Tran On Neural Networks. Nov-1991.

[5] Alvaro Sánchez. *Simulador de entornos*. Anales de mecánica y electricidad. Sept-2000.

[6] Zhang, L., B.K, Ghosth. *Line Segment Based Map Building and Localization Using 2D Laser RangeFinder",* 2000, IEEE, International Conference on Robotics and Automation. San Francisco, CA. April. pp 2538-2543.