Received: March 12, 2019. Revised: August 10, 2019. Accepted: September 7, 2019. Published: September 22, 2019.

ATNN and SVM for Autonomous Mobile Robot

A. SERRAT
LAMOSI, dept. of Computer Sciences, University Of Sciences and Technologies Oran, 31000, Algeria
serrat_amel@yahoo.fr

M. BENYETTOU
LAMOSI, dept. of Computer Sciences, University Of Sciences and Technologies Oran, 31000, Algeria
med_benyettou@yahoo.fr

Abstract—***If a robot does not know where it is, it can be difficult to determine what to do next. In order to localize itself, a robot has access to relative and absolute measurements giving the robot feedback about its driving actions and the situation of the environment around the robot. Given this information, the robot has to determine its location as accurately as possible. What makes this difficult is the existence of uncertainty in both the driving and the sensing of the robot. The uncertain information needs to be combined in an optimal way.***

The Kalman Filter is a technique from estimation theory that combines the information of different uncertain sources to obtain the values of variables of interest together with the uncertainty in these.

In this work we provide a thorough discussion of the robot localization problem resolved by Kalman Filter, Adaptive Time Delay Neural Network and Support Vector machines.

Keywords-Extended Kalman Filter, Adaptive Time Delay Neural Network, Support Vector Machines, Robot, Localization

I. INTRODUCTION

The problem of robot localization consists of answering the question Where am I? from a robot's point of view. This means the robot has to find out its location relative to the environment. When we talk about location, pose, or position we mean the x and y coordinates and heading direction of a robot in a global coordinate system.

The localization problem is an important problem. It is a key component in many successful autonomous robot systems [1]. If a robot does not know where it is relative to the environment, it is difficult to decide what to do.

The robot will most likely need to have at least some idea of where it is to be able to operate and act successfully [2, 3]. By some authors the robot localization problem has been stated as the \most fundamental problem to providing robots truly autonomous capabilities" [4].

To be able to move around in an environment a robotic vehicle has a guidance or driving system [5]. A guidance system can consist of wheels, tracks or legs, in principle anything that makes the vehicle move around. These components are called actuators.

Obviously, the guidance system plays an important role in the physical position of a robot. The guidance system directly changes the location of the vehicle. Without a guidance system the robot is not driving around, which makes localizing itself a lot easier. Assuming that a robot does have a guidance system, the way the guidance system changes the location contains valuable information in estimating the location.

II. KALMAN FILTER

The Kalman filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator that is *optimal* in the sense that it minimizes the estimated *error* covariance—when some presumed conditions are met.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: *time update* equations and *measurement update* equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate[6].

The time update equations can also be thought of as *predictor* equations, while the measurement update equations can be thought of as *corrector* equations. Indeed the final estimation algorithm resembles that of a *predictor-corrector* algorithm for solving numerical problems as shown below in Figure 1.

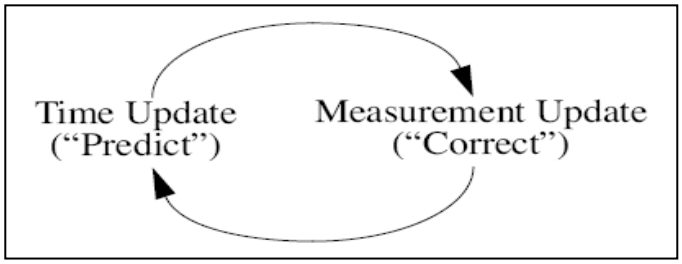


Figure 1: Kalman Filter Cycle.

The specific equations for the time and measurement updates are presented below:

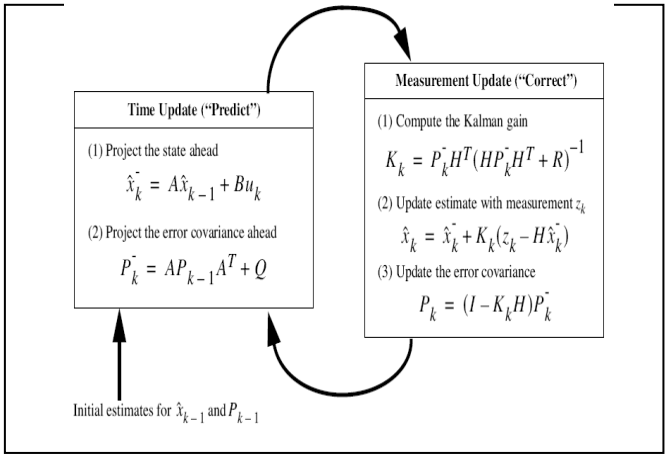


Figure 2 : Kalman Filter operations.

Where:

- *A* might change with each time step, and matrix B *relates* the optional control input *u* to the state *x*.
- *Q* is the process noise covariance; P^- is the *a priori* estimate error covariance.
- K_t is Kalman gain, *H* in the measurement equation relates the state to the measurement z_t .

III. THE EXTENDED KALMAN FILTER

The Kalman Filter above addresses the general problem of trying to estimate the state $\boldsymbol{x}$ of a discrete-time controlled process that is governed by a *linear* equation. But if the process to be estimated and (or) the measurement relationship to the process is non-linear we have work with the extended form as in figure 3.

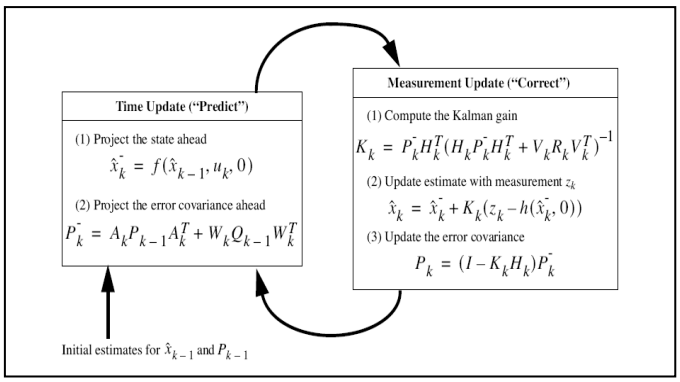


Figure 3. Extended Kalman Filter Operations.

Where

- ***H, V*** are the measurement Jacobians at step *k*, and R_k is the measurement noise covariance.

IV. ADAPTIVE TIME DELAY NEURAL NETWORK

The Adaptive Time Delay Neural Network takes the structure of the Multi Layer perceptron taking delay into account. Instead of taking all the neurons at the same time one takes only part of them. Each entry is delayed by an adjustable time delay like the weights of the network. One thus will sweep window of a fixed but variable size on the space of entry.

Figure 4 shows a delay block between two units i, j, and Figure 5 presents general view of an ATNN (Adaptive Time Delay Neural Network).

Figure 4: Delay Block diagram

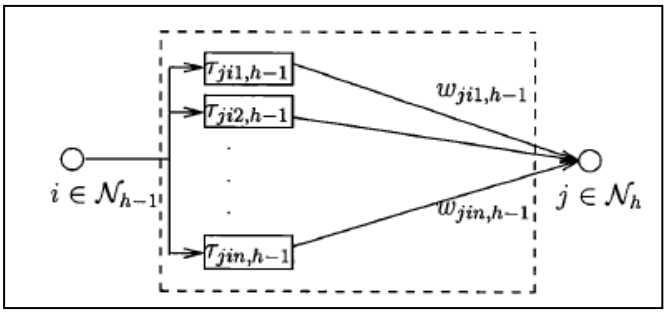
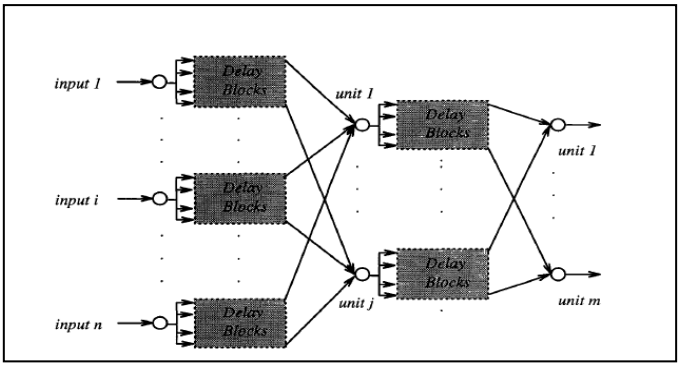


Figure 5: Diagram of ATNN.



V. TRAINING

A basic concept in the study and the use of networks of neurons is that of training. We indicate by this term, always in reference to biology, the procedure which consists in estimating the parameters of the various neurons of the network so, as well as possible; it fills the task which is affected.

It is supposed that the parameters of the network are updated in a discrete way and that the entries are differentiable.

The output of node is defined as follows [7]:

$$a_{j,h}(t_n) = \begin{cases} f_{j,h}\left(S_{j,h}(t_n)\right) & si \quad h \geq 2 \\ a_{j,0}(t_n) & si \quad h=1 \end{cases} \tag{1}$$

Where:

$$S_{j,h}(t_n) = \sum_{i \in \mathcal{N}_{h-1}} \sum_{k=1}^{K_{ji,h-1}} w_{jik,h-1} \cdot a_{i,h-1}(t_n - \tau_{jik,h-1}) \tag{2}$$

And ***f*** is a differentiable and non decreasing function definite as follows

$$f_{j,h}(x) = \frac{\beta_{j,h}}{1 + e^{-\alpha_{j,h}x}} - \gamma_{j,h} \quad (3)$$

Where

$f_{j,h}(x)$ is in $[-\gamma_{j,h}, \beta_{j,h} - \gamma_{j,h}]$.

The modification of the weights as well as the deadlines is made by the algorithm of the gradient descent. Where the error is defined as follows:

$$E(t_n) = \frac{1}{2} \sum_{j \in \mathcal{N}_L} (d_j(t_n) - a_j(t_n))^2 \quad (4)$$

Where N_L is the number of neurons in the output layer, and $d_j(tn)$ is the desired exit of the node i at time t_n.

Time Delay are adapted as follows:

$$\tau_{jik,h-1} = \tau_{jik,h-1} + \Delta\tau_{jik,h-1} \quad (5)$$

Such as:

$$\Delta\tau_{jik,h-1} = -\eta_1 \frac{\partial E(t_n)}{\partial \tau_{jik,h-1}} \quad (6)$$

The weight changes in the following way:

$$w_{jik,h-1} = w_{jik,h-1} + \Delta w_{jik,h-1} \quad (7)$$

Such as:

$$\Delta w_{jik,h-1} = -\eta_2 \frac{\partial E(tn)}{\partial w_{jik,h-1}} \quad (8)$$

VI. Support Vectors Machines

Consider the following problem: we are given a data set $g = \{(x_i, y_i)\}_{i=1}^{N}$ obtained by sampling, with noise, some unknown function $f(x)$ and we are asked to recover the function f, or an approximation of it, that has at most ε deviation from the actually obtained targets y_i for all the training data g, and at the same time is as flat as possible. In other words, we do not care about errors as long as they are less than, but will not accept any deviation larger than this [08].

$$|y - f(x)|_{\varepsilon} := \max\{0, |y - f(x)| - \varepsilon\} \quad (9)$$

A. Linear Case

We have:

$$f(x) = \langle w, x \rangle + b \qquad b \in R \quad (10)$$

Where $\langle .,. \rangle$ denotes the dot product in the space of the input patterns. ***Flatness*** in the case of (10) means that one seeks a small w. One way to ensure this is to minimize the norm. We can write this problem as a convex optimization problem [09]:

$$\min \quad \frac{1}{2}\|w\|^2 \quad (11)$$

$$\text{avec} \quad \begin{cases} y_i - \langle w, x_i \rangle - b \in \varepsilon \\ \langle w, x \rangle + b - y_i \in \varepsilon \end{cases} \quad (12)$$

The tacit assumption in (10) was that such a function actually exists that approximates all pairs (x_i, y_i) that the convex optimization problem is feasible.

Sometimes, however, this may not be the case, or we also may want to allow for some errors. One can introduce slack variables ζ_i to cope with otherwise infeasible constraints of the optimization problem (10).

We define:

$$\zeta_i = \max(0, |y_i - f(x_i)| - \varepsilon) \quad (13)$$

$$\begin{cases} \zeta_i^+ = \max(0, y_i - f(x_i) - \varepsilon) \\ \zeta_i^- = \max(0, f(x_i) - y_i - \varepsilon) \end{cases} \quad (14)$$

$$\zeta_i = \max(\zeta_i^+, \zeta_i^-) \quad (15)$$

Hence we arrive at the formulation stated in [10].

$$\min \quad \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{N} (\xi_i^+ + \xi_i^-) \quad (16)$$

$$\text{avec} \quad \begin{cases} y_i - \langle w, x_i \rangle - b - \varepsilon \leq \zeta_i^+ \\ \langle w, x_i \rangle + b - y_i - \varepsilon \leq \zeta_i^- \\ \zeta_i^+, \zeta_i^- \geq 0 \end{cases} \quad (17)$$

The constant C > 0 determines the trade-off between the flatness of f and the amount up to which ε deviations larger than are tolerated.

In most cases the optimization problem (17) can be solved more easily in its dual formulation. Moreover, the dual formulation provides the key for extending SV machine to nonlinear functions. Hence we will use a standard dualization method utilizing Lagrange multipliers, as described [10].

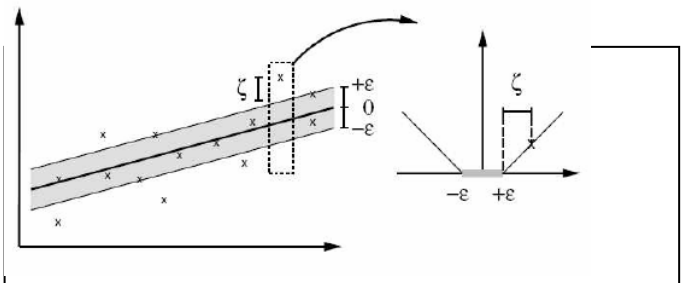


Figure 6. *The soft margin loss setting*

B. Dual problem and Qudrature Program

The idea is to build a function of Lagrange by the objective function and its constraints by introducing a dual set of variables. It can be shown that this function has a saddle point with respect to the primal and dual variables at the solution. For details see e.g. [11]. We proceed as follow:

$$L=\frac{1}{2}\|w\|^2+C\sum_{i=1}^{N}(\zeta_i^+ +\zeta_i^-)-\sum_{i=1}^{N}\alpha_i(\varepsilon+\zeta_i^+-y_i+\langle w,x_i\rangle+b)$$

$$-\sum_{i=1}^{N}\alpha_i^*(\varepsilon+\zeta_i^- +y_i-\langle w,x_i\rangle-b)-\sum_{i=1}^{N}(\eta_i\zeta_i^+ +\eta_i^*\zeta_i^-) \quad (18)$$

$$avec \quad \alpha_i,\alpha_i^*,\eta_i,\eta_i^* \geq 0$$

Where $\alpha_i,\alpha_i^*,\eta_i,\eta_i^*$ are the Lagrange Multipliers.

The Lagrangian has to be minimized with respect to ***w***, ***b*** and maximized with respect to ***α≥ 0***.

At point of optimality we have:

$$\partial_b L = \sum_{i=1}^{N}(\alpha_i-\alpha_i^*) = 0 \quad (19)$$

$$\partial_w L = w-\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)x_i = 0 \quad (20)$$

$$\partial_{\zeta_i^{(*)}} L = C-\alpha_i^{(*)}-\eta_i^{(*)} = 0 \quad (21)$$

Substituting (19, 20, and 21) in (18) we get:

$$max \quad -\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\langle x_i,x_j\rangle$$

$$-\varepsilon\sum_{i=1}^{N}(\alpha_i+\alpha_i^*)+\sum_{i=1}^{N}y_i(\alpha_i-\alpha_i^*) \quad (22)$$

$$avec \quad \begin{cases}\sum_{i=1}^{N}(\alpha_i-\alpha_i^*)=0 \\ \alpha_i,\alpha_i^* \qquad i\,[0,C]\end{cases} \quad (23)$$

In deriving (18) we already eliminated the dual variables η_i ; η_i^* through condition (21) which can be reformulated as

$$\eta_i^* = (C-\alpha_i) \quad (24)$$

Eq. (20) can be rewritten as follows:

$$w = \sum_{i=1}^{N}(\alpha_i-\alpha_i^*)x_i \quad (25)$$

Thus;

$$f(x) = \sum_{i=1}^{N}(\alpha_i-\alpha_i^*)\langle x_i,x\rangle+b \quad (26)$$

What is called support vectors expansion i.e. ***w*** can be completely described as a linear combination of the training patterns. In a sense, the complexity of a function's representation by SVs is independent of the dimensionality of the input space, and depends only on the number of SVs.

Moreover, note that the complete algorithm can be described in terms of dot products between the data. Even when evaluating ***f (x)*** we need not compute ***w*** explicitly.

C. Computing b

b can be computed by exploiting the so called Krush-Kuhn-Tucker (KKT) conditions [12]. These state that at the point of the solution the product between dual variables and constraints has to vanish.

$$\alpha_i(\varepsilon+\zeta_i^+-y_i+\langle w,x_i\rangle+b) = 0 \quad (27)$$

And

$$(C-\alpha_i)\zeta_i^+ = 0 \quad (28)$$

$$(C-\alpha_i^*)\zeta_i^- = 0 \quad (29)$$

For points which are apart from the band (ε-insensitive tubes) have

$$\alpha_i^{(*)} = C \quad (30)$$

The points inside the band have

$$\alpha_i^{(*)} = 0 \quad (31)$$

The others

$$0 < \alpha_i^{(*)} < C \quad (32)$$

The latter one has

$$\zeta = 0 \quad (33)$$

Consequently b calculation as follows [13]:

$$b = y_i-\langle w,x_i\rangle-\varepsilon \quad for \quad \alpha_i \in (0,C) \quad (34)$$

$$b = y_i-\langle w,x_i\rangle+\varepsilon \quad pour \quad \alpha_i^* \in (0,C) \quad (35)$$

From (24) it follows that only for $|y\text{-}f(x)| \geq \varepsilon$ the Lagrange multipliers may be nonzero, or in other words, for all samples

inside the ε- insensitive tube (i.e. the shaded region in Fig. 1) the vanish for $|y-f(x)|<\varepsilon$ the second factor in (27) is nonzero, hence $\alpha_i^{(*)}$ has to be zero such that the KKT conditions are satisfied. Therefore we don't need all the points to define w .The examples that come with non vanishing coefficients are called Support Vectors.

D. Linear Case Non Linear Case

This concept can be extended to the case when f is non linear. A non-linear mapping which maps the input data to a high dimensional space (also called the feature space) is introduced. We can then try to find a linear function in feature space.

Thus we avoid translating the input data to feature space first and then finding their inner products.

The difference in the linear case is that w is no longer given explicitly. Also note that in the nonlinear setting, the optimization problem corresponds to finding the flattest function in feature space, not in input space [14].

VII. Experimental Results

We consider a mobile robot with the kinematics of a unicycle. Motion is generated by two independently actuated wheels, whose rotation is measured by incremental encoders [15].

A simplified model of a mobile robot is presented in Figure 7.

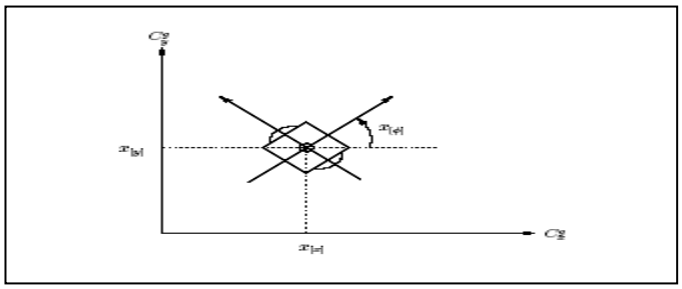


Figure 7: The Location of a Robot.

A three layered *ATNN* is used to perform the estimation task. This network contains an input layer with five inputs, one hidden layer with ten hidden units, and three output nodes to indicate the position at each step.
The number of time-delays is selected as 4 and 6 on the first and second layer of connections, respectively.
The results of ATNN are:

TABLE I. ATNN Result

		Training Error	Test error
ATNN	On X	7,97E-04	3,60E-02
	On Y	4,22E-04	3,70E-04
	On ϕ	3,10E-02	1,01E-04

And EKF results are:

TABLE II. EKF Result

Error on X	**Error on** Y	**Error on the orientation** ϕ
7.26E -04	1.73E-03	3.43E-02

And SVM results are:

TABLE III. SVM Result

	Training Error	**Test Error**
On X	1,29E-01	9.24E-04
On Y	4,29E-01	2.68E-03
On ϕ	1,17E-02	2,44E-10

According to Tables 1, 2 and 3 we see well that the new estimators proposed within the framework of this study gives better results which the EKF itself which is sensitive to the strong disturbances, and which at any moment requires the knowledge of all the variables defining the state of the system, on the other hand the SVM models and ATNN reach to the global minimum of the error.

We judge that SVM is the best estimator within the framework of our study.

VIII. Conclusion

As we see ATNN and SVM gave better results than EKF. The results show promise for using neural networks and SVM in estimating the states of a nonlinear systems.

References

[1] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," Artificial Intelligence 128,1-2, pp.99-141,2001.

[2] J. Borenstein, H. Everett, L. Feng, and D. Wehe, "Mobile robot positioning: Sensors and techniques," Journal of Robotic Systems 14, 4 , 231-249,1997.

[3] D. Kortenkamp, R. Bonasso, and R. Murphy, "AI-based Mobile Robots: Case studies of successful robot systems," MIT Press, Cambridge, MA, 1998.

[4] J. Cox J., "Blanche: Position estimation for an autonomous robot vehicle, Autonomous Mobile Robots: Control," Planning, and Architecture Vol. 2, IEEE Computer Society Press, Los Alamitos, CA, 285-292, 1991.

[5] J Cox, G. Wilfong, (Editors), "Autonomous Robot Vehicles," Springer-Verlag, New York 1990.

[6] G. Welch, G. Bishop, "An Introduction to the Kalman Filter," SIGGRAPH,2001.

[7] D.T Lin , "Trajectory production with the adaptive time-delay neural network," Neural network, Vol 8,3, pp. 447-46 I, 1994.

[8] K. P. Bennett and O. L Mangasarian.:«Robust linear programming discrimination of two linearly inseparable sets». Optimization Methods and Software, 1992, pp 1,23-34.

[9] H. W. Kuhn and A. W.: Tucker «Nonlinear programming». In Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics, Berkeley, University of California Press. 1951 pp 481-492.

[10] G. P. McCormick «Nonlinear Programming: Theory, Algorithms, and Applications». John Wiley and Sons, New York, 1983.

[11] L.Miclet « Apprentissage Artificiel : Méthodes et Algorithmes ». Eyrolles, Novembre, 2002.

[12] A.J. Smola and B. Scholkopf « A Tutorial on Support Vectors regression». NeuroColt2, 1998.

[13] V Vapnik.: « The Nature of Statistical Learning Theory », Springer, New York, 1995.

[14] K. P. Bennett and O. L Mangasarian.:«Robust linear programming discrimination of two linearly inseparable sets». Optimization Methods and Software, 1992, pp 1,23-34.

[15] R Negenborn, "Robot Localization and Kalman Filters" M.S. thesis, Utrecht -University, 2003.