Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Applying SOM as a Search Mechanism for Dynamic System

Yi-Yuan Chen and Kuu-Young Young
Department of Electrical and Control Engineering
National Chiao-Tung University, Hsinchu, Taiwan
National Chiao-Tung University Vision Research Center

Abstract— **The self-organizing map (SOM), as a kind of unsupervised neural network, has been applied for both static data management and dynamic data analysis. To further exploit its ability in search, in this paper, we employ the SOM as a searching mechanism for dynamic system. A learning scheme, consisting mainly of the SOM and the target dynamic system, is then proposed. The performance of this SOM-based learning scheme is especially compared with that of the genetic algorithm (GA) due to their resemblance in learning and searching. And, a new SOM weight updating rule is proposed to enhance learning efficiency, which may dynamically adjust the neighborhood function for the SOM in learning system parameters. For demonstration, the proposed learning scheme is applied for trajectory prediction, and its effectiveness evaluated via the simulations based on using the SOM, GA, and also Kalman filtering.**

Key words: Self-Organizing Map, Dynamic System, Genetic Algorithm, Trajectory Prediction.

I. INTRODUCTION

The self-organizing map (SOM), as a kind of unsupervised neural network, is performed in a self-organized manner in that no external teacher or critic is required to guide synaptic changes in the network [3], [9]. By contrast, for the other two basic learning paradigms in neural network, the supervised learning is performed under the supervision of an external teacher [7], and reinforcement learning involves the use of a critic that evolves through a trial-and-error process [2]; both of them demand the input-output pairs as the training data. The appealing features in learning without the input-output pairs makes the SOM very attractive in dealing with varying and uncertain data. In its many applications, the SOM has been used for static data management, such as data mining, knowledge discovering, clustering, visualization, and robot control [7], [8], [9], [10], [15], and also dynamic data analysis, such as local dynamic modeling and moving object tracking [1], [12], [14]. However, from our survey, its ability in search has not been well exploited yet [6], [11]. It thus motivates us to propose an SOM-based learning scheme for search of dynamic system, a novel employment of the SOM.

For this SOM application on search of dynamic system, the goal may be to look for a set of optimal parameters that lead to the desired performance of the dynamic system from some measured data. For instance, in a missile interception application, the task may be to predict the most probable launching position and velocity of an incoming missile from the measured radar data. Thus, the proposed SOM-based learning system should be able to evaluate system performance and execute the subsequent search. In developing the system, we will first examine the SOM in its learning strategy and effectiveness. The genetic algorithm (GA) is especially taken for comparison, since these two learning schemes exhibit some resemblance in searching. Both the search domains in the SOM and GA are evolving, leading to more concentrated searches, until a satisfactory solution is found. The major difference lies in the mechanism that makes the search domain evolve. Meanwhile, as a search mechanism for dynamic system, the SOM learning may involve both the system parameters and their derivatives, which operate in quite different ranges. To achieve high learning efficiency under such a wide parameter variation, we propose a new weight updating rule, which may dynamically adjust the shape of the neighborhood function for the SOM, in an individual basis, in learning the system parameters. The rest of this paper is organized as follows. The proposed SOM-based learning scheme and its comparison with the GA are discussed in Sec. 2. Its system implementation and application on trajectory prediction are described in Sec. 3. In Sec. 4, simulations are conducted to compare the performance of the SOM with those of the GA and Kalman filtering. Finally, conclusions are given in Sec. 5.

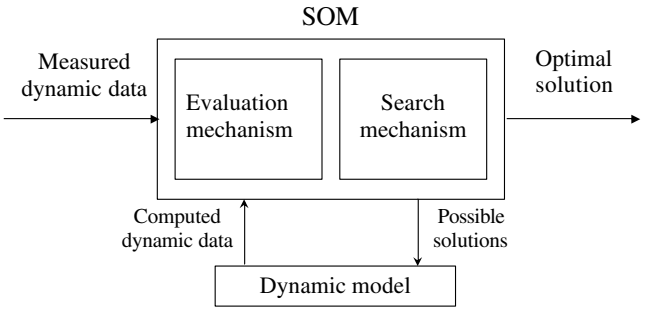


Fig. 1. Proposed SOM-based learning scheme.

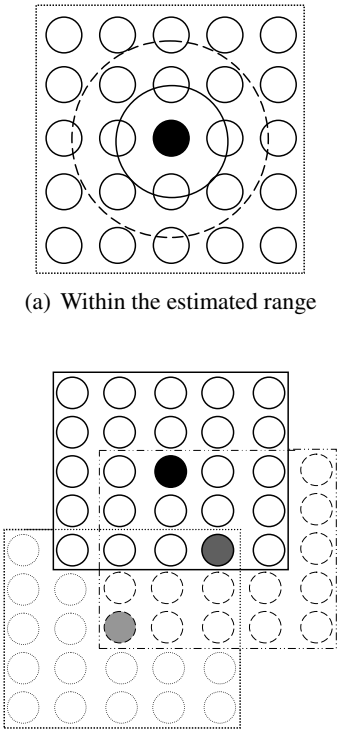
II. PROPOSED SOM-BASED LEARNING SCHEME

Figure 1 shows the conceptual diagram of the proposed SOM-based learning scheme. In Figure 1, the SOM consists of mainly the evaluation and search mechanisms, and the dynamic model stands for the target system. Initially, the function for performance evaluation will be installed in the evaluation mechanism, and possible solutions (e.g., vectors of system parameters), selected from an estimated range, distributed among the neurons of the SOM. During each time interval of the learning process, all the possible solutions in

the neurons will be sent to the dynamic model to derive the dynamic data that illustrate their performance. The evaluation mechanism will then compute the difference between the derived dynamic data and the incoming measured dynamic data. From the results, the search mechanism chooses the solution leading to the most accurate derived dynamic data as the winner, and updates the weights of this winner and its neighboring neurons. The learning process then continues, and the network will eventually converge to the optimal solution. To note that, even the optimal solution is not within the estimated range for some cases, the search mechanism is still expected to move the possible candidates out of their initial locations and guide them to converge to the optimal solution. From the process described above, the SOM employed in this way is very similar to the GA in searching; meanwhile, they adopt different search mechanism and paradigm. It is of interest how their choices influence their performance.

For comparison, we start with the examination of the SOM and GA in their structures and operations. The SOM, first introduced by Kohonen, transforms input vectors into a discrete map (e.g., a 2-D grid of neurons) in a topological ordered fashion adaptively [9], [14]. The SOM is with a structure very suitable for parallel processing. We further exploit this parallelism and design an organized search accordingly. In other words, we take advantage of the SOM in its distribution of the neurons in a grid pattern and the presence of local interaction in between the grid. Take the missile interception application as an example again. When the estimated ranges of the possible launching position and velocity of the missile are available, we may distribute the possible positions and velocities into the network in an organized fashion. Under this arrangement, the searches among the neurons are closely related through the grid, leading to a more rapid convergence. On the other hand, when the estimation is inaccurate, the search, still organized, may take longer time to converge to the optimal solution. For illustration, Figure 2 shows the conceptual diagram of the organized search in a 2-D SOM. Figure 2(a) shows a case that the solution is within the estimated range. In this case, the weights of the neurons are updated so as to make the weight vectors converge to a compact cluster centering at the optimal solution. And, Figure 2(b) shows the case that the solution is outside of the estimated range, in which the weights are updated in a manner that moves these weight vectors toward the optimal solution located outside of the estimated range.

The GA, first introduced by Holland, has many successful applications in various areas [4], [6]. Basically, the GA is a search algorithm based on the mechanics of natural selection and natural genetics. It employs multiple concurrent search points called chromosomes and evaluates the fitness of each chromosome. The search procedure uses random choice as a tool to guide a highly exploitative search through a coding of a parameter space. The GA in general consists of three operators: reproduction, crossover, and mutation. To achieve effective search, the GA uses the crossover operator to unite an individual which is doing well in one dimension with some other individual that is doing well in another dimension. The effect of reproduction and crossover operators are like casting nets from locations in the search space, which are occupied by individuals that have good fitness. While reproduction and crossover can cover much of the search space, occasionally they may lose some potentially useful genetic material. The mutation operator provides a new location where the net has never been and protects against such an irrecoverable loss. By itself, mutation is a random walk through the search space. When used sparingly with reproduction and crossover, it is an insurance policy against premature loss of important material.



(a) Within the estimated range

(b) Outside of the estimated range

Fig. 2. Conceptual diagram of the organized search in a 2-D SOM: the solution is (a) within the estimated range and (b) outside of the estimated range.

Based on the discussions above, both the SOM and GA have the merit of parallel processing. And, both of their searches are through the guidance of the evaluation function, while the SOM in our design adopts a somewhat organized search and the GA in some sense a random approach. It implicates that the SOM may be more suitable for applications with certain knowledge, especially when the distribution of the possible solutions is not utterly random. On the contrary, for applications with no a priori knowledge available, the GA may yield better performance. Later in Sec. 4, an application on trajectory prediction is used to evaluate their effectiveness.

III. System Implementation

Figure 3 shows the structure and operation of the SOM in the proposed scheme. This proposed SOM performs two operations: evaluation and search. In Figure 3, each neuron

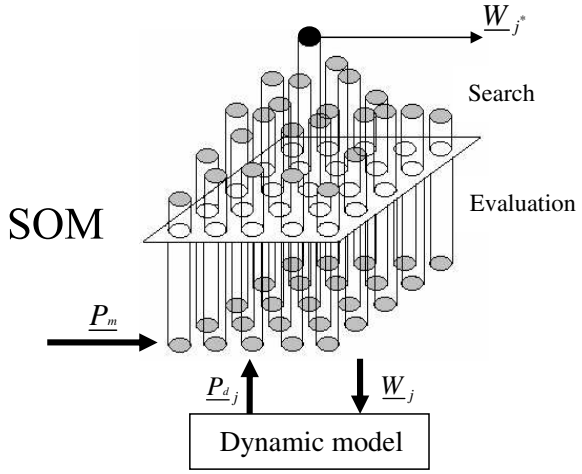


Fig. 3. The structure and operation of the SOM in the proposed scheme.

j in the SOM contains a vector of a possible solution set $\underline{W}_j$ (the weight vector). Each time new measured dynamic data $\underline{P_m}$ are sent into the scheme, the SOM is triggered to operate. All of the possible solution sets in the neurons will then be sent to the dynamic model to derive their corresponding dynamic data $\underline{P_{d\,j}}$. The SOM evaluates the difference between $\underline{P_m}$ and each $\underline{P_{d\,j}}$. Of all the neurons, it chooses the neuron j^*, which corresponds to the smallest difference, as the winner. When the weight vector $\underline{W}_{j^*}$ of this winning neuron j^* differs from $\underline{W}_{\hat{j}^*}$ of the previous winner $\hat{j}^*$, the weight vectors of $\hat{j}^*$ and its neighbors will be updated in a manner that moves these weight vectors toward neuron j^*. When j^* is the same as $\hat{j}^*$, the weight vectors will be updated so as to make them form more and more compact clusters centering at neuron j^*. Under successful learning, the SOM will finally converge to an optimal solution set.

Several parameters need to be determined for weight updating in the SOM, including the topological neighborhood function and learning rate. Their determination may depend on the properties of the system parameters to learn. As mentioned above, system parameters (including their derivatives) may operate in quite different working ranges. To achieve high learning efficiency, the weight updating should be executed in an individual basis, instead of using a same neighborhood function for all the parameters. We thus propose a new SOM weight updating rule, which can dynamically adjust the center and width of their respective neighborhood function for the SOM in learning each of the system parameters. The proposed weight updating rule is designed to make the distance between neuron j and j^* correspond to that of their weight vectors, $\underline{W}_j$ and $\underline{W}_{j^*}$. From an individual basis, we first define two Gaussian neighborhood functions $D(j(k))$ and $F(w_{j,i}(k))$ for the parameter $w_{j,i}(k)$ (the ith element in $\underline{W}_j(k)$) in the kth stage of learning as

$$D(j(k)) = \exp(-\frac{d^2_{j,j^*}(k)}{2\sigma_d^2}) \quad (1)$$

$$F(w_{j,i}(k)) = \exp(-\frac{(w_{j,i}(k)-w_{j^*,i}(k))^2}{2\sigma^2_{w_i}}) \quad (2)$$

where $d_{j,j^*}(k)$ stands for the lateral connection distance between neuron j and j^*, $w_{j^*,i}(k)$ the ith element in $\underline{W}_{j^*}(k)$, and σ_d and σ_{w_i} the standard deviation of the neighborhood function $D(j(k))$ and $F(w_{j,i}(k))$, respectively. An error function $E_1(k)$ is then defined as

$$E_1(k) = \frac{1}{2}(D(j(k)) - F(w_{j,i}(k)))^2 \quad (3)$$

To speed up the learning, we propose varying the mean and variance of the neighborhood function $F(w_{j,i}(k))$ by moving its center to where $w_{j^*,i}(k)$ is located and enlarging (reducing) the variance $\sigma^2_{w_i}$ to be ${\sigma^n_{w_i}}^2 = |w_{j^*,i}(k) - \tilde{w}_i(k)|^2$, where $\tilde{w}_i(k)$ stands for the average of all $w_{j,i}(k)$ in the neurons and $|\cdot|$ the difference. The new neighborhood function $F^n(w_{j,i}(k))$ is then formulated as

$$F^n(w_{j,i}(k)) = \exp(-\frac{(w^n_{j,i}(k)-w_{j^*,i}(k))^2}{2{\sigma^n_{w_i}}^2}) = \exp(-\frac{(w_{j,i}(k)-\tilde{w}_i(k))^2}{2\sigma^2_{w_i}}) \quad (4)$$

where $w^n_{j,i}(k)$ is the new weight value for $w_{j,i}(k)$ after the variation. Under this design, during each iteration of learning, $F^n(w_{j,i}(k))$ is dynamically centered at the winning neuron j^*, with a larger (smaller) width when $\tilde{w}_i(k)$ is more (less) different from $w_{j^*,i}(k)$, thus covering a more fitting neighborhood region. Higher learning efficiency can then be expected. With $F^n(w_{j,i}(k))$, the new weight $w^n_{j,i}(k)$ is derived as

$$w^n_{j,i}(k) = \frac{|w_{j^*,i}(k) - \tilde{w}_i(k)|}{\sigma_{w_i}} \cdot (w_{j,i}(k) - \tilde{w}_i(k)) + w_{j^*,i}(k) \quad (5)$$

With a desired new weight $w^n_{j,i}(k)$, in addition to minimizing the error function $E_1(k)$ in Eq.(3), the learning should also make $w_{j,i}(k)$ approach $w^n_{j,i}(k)$. A new error function $E(k)$ is thus defined as

$$E(k) = \frac{1}{2}[(D(j(k)) - F(w_{j,i}(k)))^2 + (w_{j,i}(k) - w^n_{j,i}(k))^2] \quad (6)$$

Based on the gradient-descent approach, the weight-updating rule is derived as

$$\begin{aligned} w_{j,i}(k+1) &= w_{j,i}(k) - \eta(k)\frac{\partial E(k)}{\partial w_{j,i}(k)} \\ &= w_{j,i}(k) - \eta(k)[\frac{\partial E_1(k)}{\partial F(w_{j,i}(k))} \cdot \frac{\partial F(w_{j,i}(k))}{\partial w_{j,i}(k)} + \\ &\quad (w_{j,i}(k) - w^n_{j,i}(k))] \\ &= w_{j,i}(k) - \eta(k)[\frac{(w_{j,i}(k)-w_{j^*,i}(k))}{\sigma^2_{w_i}} \cdot F(w_{j,i}(k)) \cdot (D(j(k)) \\ &\quad - F(w_{j,i}(k))) + (w_{j,i}(k) - w^n_{j,i}(k))] \end{aligned} \quad (7)$$

where $\eta(k)$ stands for the learning rate in the kth stage of learning. The selection of $\eta(k)$ should depend on the closeness between $w_{j,i}(k)$ and $w^*_{j,i}(k)$. When they are different

from each other, the learning process can be speeded up with a larger $\eta(k)$. And when they almost coincide, the learning rate may be decreased gradually. A function for $\eta(k)$ that satisfies the demand is formulated as

$$\eta(k) = \eta_1 \cdot e^{-k/\tau} + \eta_0 \tag{8}$$

where η_0 and η_1 are constants smaller than 1, and τ time constant. Of course, other types of functions can also be used. Together, the weight updating rule described in Eq.(7) and the learning rate in Eq.(8) will force the minimization of the difference between the weight vector of the winning neuron and those corresponding to every neuron in each learning cycle. The learning will converge eventually.

A. Trajectory prediction application

To evaluate its effectiveness, we apply the proposed SOM-based learning scheme for a trajectory prediction task. As a trajectory predictor, the scheme is used to estimate the launching position and velocity of a moving object using the measured data. Through a learning process, the predictor may determine a most probable initial state through repeatedly comparing the measured data with the predicted trajectories derived from the possible initial states stored in the neurons of the SOM. We consider the proposed scheme suitable for this application, because the relationship between the initial state and its resultant trajectory is not utterly random. We can thus distribute the initial states into the SOM in an organized fashion, and make it a guided search.

In this application, the nonlinear dynamic equation describing the trajectory of the moving object and the measurement equation are first formulated as

$$\underline{x}(k+1) = f_k(\underline{x}(k)) + \underline{\xi}_k \tag{9}$$

$$\underline{p}(k) = g_k(\underline{x}(k)) + \underline{\zeta}_k \tag{10}$$

where f_k and g_k are the vector-value function defined in R^l and R^q (l and q the dimension), respectively, and their first-order partial derivatives with respect to all the elements of $\underline{x}(k)$ continuous. $\underline{\xi}_k$ and $\underline{\zeta}_k$ are the zero-mean Gaussian white noise sequence in R^l and R^q, respectively, with

$$E[\underline{\xi}_k] = 0 \tag{11}$$

$$E[\underline{\xi}_j \underline{\xi}_k^T] = Q\delta_{jk} \tag{12}$$

$$E[\underline{\zeta}_k] = 0 \tag{13}$$

$$E[\underline{\zeta}_j \underline{\zeta}_k^T] = R\delta_{jk} \tag{14}$$

$$E[\underline{\xi}_j \underline{\zeta}_k^T] = 0 \tag{15}$$

where $E[\cdot]$ stands for the expectation function, Q and R the covariance matrix of the input noise and output noise, respectively, and δ_{jk} the Dirac delta function. Q and R are expected to be uncertain and varying in noisy, unknown environments, and their estimated values possibly imprecise, even incorrect. Being unaware of the statistical properties of the dynamic model, the proposed scheme is utilized to find the optimal initial state via learning. According to the description of the proposed scheme in Sec. 3.1 and the dynamic model and measurement equation in Eqs.(9)-(10), the SOM-based learning algorithm for trajectory prediction is organized as follows:

SOM-based Learning Algorithm for Trajectory Prediction: Predict an optimal initial state for the trajectory of a moving object using the measured position data.

Step 1: Set the stage of learning $k = 0$. Estimate the ranges of the possible launching position and velocity of the moving object, and randomly store the possible initial states $\underline{W}_j(0)$ into the neurons, where $j = 1, \ldots, m \times n$, $m \times n$ the total number of neurons in the 2D ($m \times n$) space.

Step 2: Send $\underline{W}_j(k)$ into the dynamic model, described in Eq.(9), to compute $\underline{P_d}_j(k)$.

Step 3: For each neuron j, compute its output $O_j(k)$ as the Euclidean distance between the measured position data $\underline{P_m}(k)$ and $\underline{P_d}_j(k)$:

$$O_j(k) = \|\underline{P_m}(k) - \underline{P_d}_j(k)\| \tag{16}$$

Find the winning neuron j^* with the minimum $O_{j^*}(k)$:

$$O_{j^*}(k) = \|\underline{P_m}(k) - \underline{P_d}_{j^*}(k)\| = \min_j \|\underline{P_m}(k) - \underline{P_d}_j(k)\| \tag{17}$$

Step 4: Update the weight vectors of the winning neuron j^* and its neighbors using Eq.(7).

Step 5: Check whether all the differences between each $w_{j^*,i}(k)$ of the winning neuron j^* and those $w_{j,i}(k)$ corresponding to every neuron j are smaller than a pre-specified value ϵ:

$$\max_{j,i} |w_{j,i}(k) - w_{j^*,i}(k)| < \epsilon \tag{18}$$

If Eq.(18) does not hold, let $k = k + 1$ and go to Step 2; otherwise, the prediction process is completed and output the predicted optimal initial state to the dynamic model to derive the object trajectory. Note that the value of ϵ is empirical according to the demanded resolution in learning, and we chose it very close to zero. In addition, during each stage of learning, we perform a number of learning to increase the SOM learning speed. This number of learning is set to be a large number in the initial stage of the learning process, such that the SOM may converge faster at the price of more oscillations, and decreased gradually to achieve smooth learning in the later stages of learning.

IV. Simulation

For demonstration, we performed a series of simulations for trajectory prediction based on using the SOM, GA, and Kalman filter. Kalman filtering is a famous approach for trajectory prediction, which has been widely used in predicting the movements of the satellites, airplanes, ships, etc. [5], [13]. In the simulations, the trajectory to predict was for a moving object that emulated a missile. Its governing equations of motion in the 3D Cartesian coordinate system are described as

$$\ddot{x} = \frac{-g_m x}{(x^2+y^2+z^2)^{3/2}} + 2\omega\dot{y} + \omega^2 x + \xi_x \quad (19)$$

$$\ddot{y} = \frac{-g_m y}{(x^2+y^2+z^2)^{3/2}} + 2\omega\dot{x} + \omega^2 y + \xi_y \quad (20)$$

$$\ddot{z} = \frac{-g_m z}{(x^2+y^2+z^2)^{3/2}} + \xi_z \quad (21)$$

where g_m and ω stand for the gravitational constant and the rotative velocity of the earth, respectively, and set to be $g_m = 3.986 \times 10^5 km^3/s^2$ and $\omega = 7.2722 \times 10^{-5} rad/s$. (ξ_x, ξ_y, ξ_z) are assumed to be continuous-time uncorrelated zero-mean Gaussian white noise processes. Referring to Eq.(9) and letting $\underline{X} = (x, y, z, \dot{x}, \dot{y}, \dot{z})^T = (x_1, x_2, x_3, x_4, x_5, x_6)^T$, we can obtain the discretized dynamic equation as

$$\underline{X}(k+1) = f(\underline{X}(k)) + \underline{\xi}_k \quad (22)$$

where

$$f(\underline{X}(k)) = \begin{bmatrix} x_1(k)+tx_4(k) \\ x_2(k)+tx_5(k) \\ x_3(k)+tx_6(k) \\ x_4(k)-tg_m x_1(k)/(x_1(k)^2+x_2(k)^2+x_3(k)^2)^{3/2}+ \\ 2t\omega x_5(k)+t\omega^2 x_1(k) \\ x_5(k)-tg_m x_2(k)/(x_1(k)^2+x_2(k)^2+x_3(k)^2)^{3/2}+ \\ 2t\omega x_4(k)+t\omega^2 x_2(k) \\ x_6(k)-tg_m x_3(k)/(x_1(k)^2+x_2(k)^2+x_3(k)^2)^{3/2} \end{bmatrix} \quad (23)$$

and

$$\underline{\xi}_k = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \xi_{x_4} \\ \xi_{x_5} \\ \xi_{x_6} \end{bmatrix} \quad (24)$$

with t the sampling time. $(\xi_{x_4}, \xi_{x_5}, \xi_{x_6})$ are assumed to be uncorrelated zero-mean Gaussian white noise sequences with a constant variance $\sigma_f^2 = (0.1m/s^2)^2$. And, referring to Eq.(10), the measurement equation is formulated as

$$\underline{P}(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \underline{X}(k) + \underline{\zeta}_k \quad (25)$$

where $\underline{\zeta}_k = (\zeta_{x_1}, \zeta_{x_2}, \zeta_{x_3})^T$ are the measurement noise sequences with a zero mean and constant variance $\sigma_m^2 = (15m)^2$. The ranges of the possible initial states $\underline{W}_j(0)$ were estimated to be

$$\begin{array}{l} 68.6 \times 10^5 m \le x_1(0) \le 68.8 \times 10^5 m \\ 2.7 \times 10^5 m \le x_2(0) \le 2.8 \times 10^5 m \\ 4.8 \times 10^5 m \le x_3(0) \le 4.9 \times 10^5 m \\ 110 m/s \le x_4(0) \le 150 m/s \\ 810 m/s \le x_5(0) \le 850 m/s \\ 1360 m/s \le x_6(0) \le 1380 m/s. \end{array} \quad (26)$$

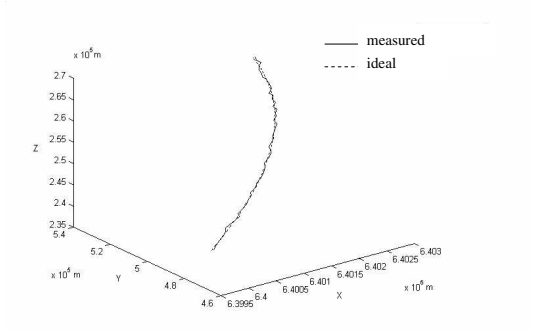
Within the ranges described in Eq.(26), the possible launching positions and velocities of the missile were selected and stored into the 729 (27×27) neurons of the 2D SOM. The learning rates for each parameter were determined according to Eq.(8), with the time constant τ set to be 100 and η_0 and η_1 between 0 and 0.8. The sampling time t was $0.5s$. For the GA, the population size was selected to be 729 to match with the SOM, and the crossover and mutation probability 0.6 and 0.0333, respectively. As for the Kalman filter, we randomly chose 500 possible initial states within the ranges to estimate an initial state $X_0 = E[X(0)]$ and an error covariance matrix as $E[(X(0) - X_0)(X(0) - X_0)^T]$.

We applied the SOM, GA, and Kalman filter for trajectory prediction under the following four situations: (1) good estimates of both the initial state and noise distribution, (2) good estimate of the initial state, but bad estimate of the noise distribution, (3) bad estimate of the initial state, but good estimate of the noise distribution, and (4) bad estimates of both the initial state and noise distribution. From the simulation results for Case 1, all the SOM, GA, and Kalman filter predicted the initial state quite well and thus resulted in very small estimated errors, except in the initial stage of the prediction. For Cases 2-4, the performance of the Kalman filter degraded with bad estimate in either noise distribution or initial state. The influence of bad estimate on the SOM was mostly at the initial stage of the prediction. After the transient, the SOM still managed to find the optimal initial state. As for the GA, it converged very slowly when the optimal initial state did not fall within the estimated range. And, it was not that straightforward to determine a proper population size and crossover and mutation probabilities to speed up its convergence rate. We thus conclude that the proposed SOM-based learning scheme performed better than the GA and Kalman filter in robustness and efficiency for this trajectory prediction application.

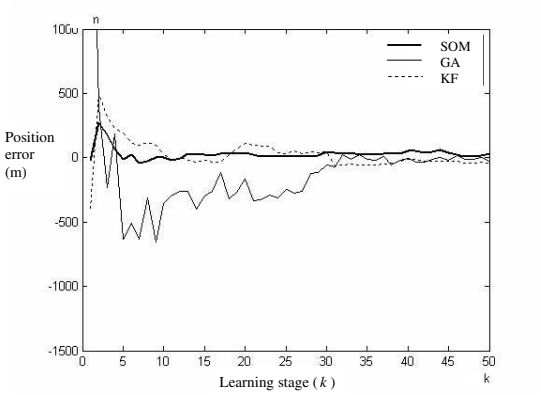
The simulation results for Case 4 are shown in Figure 4 for illustration. In this simulation, the ideal initial state was assumed to be $(64 \times 10^5 m, 4.8 \times 10^5 m, 2.4 \times 10^5 m, 215 m/s, 2130 m/s, 1030 m/s)$, which was outside the estimated range. And, the variance of the measurement noise was enlarged to be $(30m)^2$. Figure 4(a) shows the ideal and measured trajectories, Figure 4(b) the estimated position error, and Figure 4(c) the variation of the neighborhood function $F(w_{j,1}(k))$ during the SOM learning process. In Figure 4(c), the center of $F(w_{j,1}(k))$ leaned toward the right side of the figure, because the optimal solution was located outside of the estimated range.

V. Conclusion

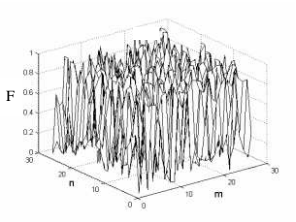
In this paper, we have proposed an SOM-based learning scheme, in which the SOM is used as a search mechanism for dynamic system. Via the examination of the SOM and GA in their structures and learning strategies, we consider that the SOM is more suitable than the GA for applications with certain knowledge of the possible solutions. To achieve high efficiency in learning a number of dynamic parameters, we have also proposed a new SOM weight updating rule.



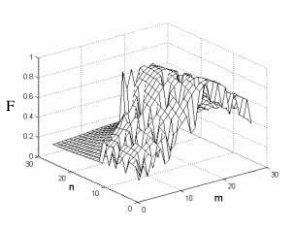
(a) Ideal and measured trajectories



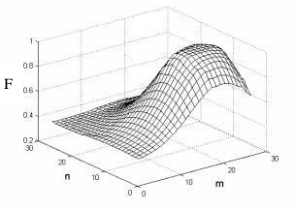
(b) Estimated position error in the X-direction



$k = 0$



$k = 25$



$k = 50$

(c) Variation of the neighborhood function $F(w_{j,1}(k))$

Fig. 4. Simulation results for trajectory prediction using the SOM, GA, and Kalman filter with bad estimates of both the initial state and noise distribution: (a) the ideal and measured trajectories, (b) the estimated position error in the X-direction, and (c) the variation of the neighborhood function $F(w_{j,1}(k))$ during the SOM learning process.

The performance of the proposed scheme has been compared with those of the GA and Kalman filter via the simulations for a trajectory prediction task. To further exploit its search ability, in future work, we will apply the SOM for dynamic optimization problem, with the issue on persistent excitation investigated [16]. As the SOM also possesses an appealing feature in responding to distinct properties exhibited by input data through forming several corresponding clusters, another worthwhile future work will be to extend the proposed scheme for systems involving multiple targets.

VI. Acknowledgment

This work was supported in part by the National Science Council under grant NSC 93-2218-E-009-061, and also Department of Industrial Technology under grant 93-EC-17-A-02-S1-032.

References

[1] G. A. Barreto and A. F. R. Araujo, "Identification and Control of Dynamical Systems Using the Self-Organizing Map," *IEEE Trans. on Neural Networks*, Vol. 15(5), pp. 1244-1259, 2004.

[2] A. G. Barto, "Reinforcement Learning and Adaptive Critic Methods," *Handbook of Intelligent Control*, White and Sofge, eds., Van Nostrand-Reinhold, New York, pp. 469-491, 1992.

[3] G. A. Carpenter and S. Grossberg, "The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network," IEEE Computer, Vol. 21(3), pp. 77-88, 1988.

[4] Y. Davidor, *Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization*, World Scientific, New Jersey, 1991.

[5] M. Efe and D. P. Atherton, "Maneuvering Target Tracking with an Adaptive Kalman Filter," *IEEE Conference on Decision and Control*, pp. 737-742, 1998.

[6] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, New York, 1989.

[7] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan, New York, 1994.

[8] H.-D. Jin, K.-S. Leung, M.-L. Wong, and Z.-B. Xu,"An Efficient Self-Organization Map Designed by Genetic Algorithms for the Traveling Salesman Problem," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 33(6), pp. 877-888, 2003.

[9] T. Kohonen, *Self-Organizing Map*, Springer, Berlin, Germany, 1997.

[10] T. M. Martinetz, H. J. Ritter, and K. J. Schulten, "Three-Dimensional Neural Net for Learning Visuomotor Coordination of a Robot Arm," *IEEE Trans. on Neural Networks*, Vol. 1(1), pp. 131-136, 1990.

[11] K. Obermayer and T. J. Sejnowski, ed., *Self-Organizing Map Formation: Foundation of Neural Computation*, MIT Press, Cambridge, 2001.

[12] J. C. Principe, L. Wang, and M. A. Motter, "Local Dynamic Modeling with Self-Organizing Maps and Applications to Nonlinear System Identification and Control," *Proceedings of the IEEE*, Vol. 86(11), pp. 2240-2258, 1998.

[13] K. V. Ramachandra, "A Kalman Tracking Filter for Estimating Position, Velocity and Acceleration from Noisy Measurements of a 3-D Radar," *Electro Technology*, Vol. 33, pp. 66-76, 1989.

[14] H. Shah-Hosseini and R. safabakhsh, "TASOM: a New Adaptive Self-Organization Map," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 33(2), pp. 271-282, 2003.

[15] M. C. Su and H. T. Chang, "Fast Self-Organizing Feature Map Algorithm," *IEEE Trans. on Nueral Networks*, Vol. 11(3), pp. 721-733, 2000.

[16] M. C. Su, Y. X. Zhao, and J. Lee, "Som-based Optimization," *IEEE Int. Conference on Neural Networks*, pp. 781-786, 2004.