2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Improved Training of Cellular SRN Using Unscented Kalman Filtering for ADP

L. Vidyaratne[1], M. Alam[1], J. K. Anderson[2], and K. M. Iftekharuddin[1]

***Abstract*—Cellular Simultaneous Recurrent Network (CSRN) is a unique type of recurrent networks that is designed to solve complex optimization problems. This network has already shown to successfully solve many challenging problems such as 2D maze navigation, image registration and affine transformation, game of go, and power system voltage profile prediction. One of the main challenges of using a complex network structure as CSRN is to efficiently train the network. Many representative training algorithms such as Back-propagation Through Time (BPTT), Extended Kalman Filtering (EKF) and Particle Swarm Optimization (PSO) have been used to train CSRN. Our prior works with CSRN suggest that for large number of network inputs, which is very common for large scale maze and image data, computational complexity of computing Jacobian in EKF training becomes prohibitive. In this paper, we propose Unscented Kalman Filter (UKF) for the training of CSRN to avoid computing Jacobian. We show that CSRN trained with UKF can solve the 2D maze traversal problem with better convergence rate than that of EKF. We also report preliminary results on binary image affine transformation wherein CSRN trained with UKF offers comparable performance to that of EKF. A comparison has been obtained between CSRN with GMLP core versus an Elman core trained with UKF for Affine transform results. Finally, we show that for more complex applications such as large scale image processing, UKF is much faster than EKF in training CSRN.**

I. Introduction

Artificial Neural Networks (ANNs) are inspired by human and animal neural pathways such that the ANNs can mimic the ability to learn and adapt. One of the main purposes of ANNs is to learn various nonlinear functions that are used in different applications. Among many widely used ANNs, Feed-forward neural networks are considered as universal function approximators. However, in many practical applications with large number of input variables, function approximation may not always be effective with generic feed-forward networks since the required approximation complexity increases exponentially. In order to solve nonlinear functions, more complex networks are needed. Cellular Simultaneous Recurrent Network (CSRN) is such a network which has exhibited the ability to solve complex optimization problems [1] [2] [3] [4]. Although CSRN has the ability to approximate complex functions more successfully compared to that of feed-forward or multilayer perceptron (MLP) networks [1], the complexity of the network itself makes training a challenging task.

CSRN was first introduced by Pang et al. [1], wherein the authors solved 2D maze traversal problem. The authors have shown that the solution of 2D maze navigation problem is similar to approximating the *J* function of adaptive dynamic programming (ADP). The idea behind ADP is to approximate the exact solution of Bellman's optimality equation given below,

$$J(i) = \min\left(c(i,\mu(i)) + \gamma \sum_{j=1}^{N} B_{ij}(\mu) J(j)\right) \qquad (1)$$

where, the total estimated cost from the starting state "*i*" is *J(i)*, γ is the discount factor and μ is the policy, N is the number of possible states, B_{ij} indicates the probability, c(i, j) is the expected cost between any two states *"i"* and *"j"*. Optimal policy offers the optimal estimated cost [5]. The authors in [1] use Back Propagation Through Time algorithm (BPTT) to train the network. A single maze is trained with the CSRN and tested with the same maze. The paper shows that CSRN is successful to approximate the *J* function. In [5], the authors later show that CSRN can also be more effectively trained with Extended Kalman Filter (EKF) algorithm. Here, the authors propose that CSRN can be trained with a different set of mazes and Fthe network can learn any random maze. The authors also show that the use of EKF algorithm rather than BPTT improves the speed of convergence by several orders.

The BPTT is an extension of the standard back-propagation algorithm that enables training neural networks with feed-back connections. Hence, BPTT 'unfolds' the recurrent networks through a certain number of iterations. This forms a feed-forward network consisting replications of the original network arranged in layers with the feedback connections now feeding forward to the replication in front. Once the network is 'unfolded', standard back-propagation is applied for training. It is shown that in Pang et al. [1] that CSRN trained by BPTT for 2D maze traversal converged with approximately 1000 epochs.

Kalman filters provide computational means to estimate the state of a linear system using series of past observations. The EKF is a generalization of the Kalman filter which can be used for nonlinear system estimation. EKF essentially 'linearizes' the system by computing the derivatives (Jacobian) of the nonlinear function prior to applying the

[1] LV, MA and KMI are with the Vision Lab at Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23529 (email:{lvidy001, malam001, kiftekha}@odu.edu)

[2]JKA, was with Department of Electrical and Computer Engineering, University of Memphis, Memphis, TN 38018, when part of this work was done. (e-mail: keith.anderson@thyssenkrupp.com).

This work is partially supported through a grant funding from NSF (Award #1310353).

standard Kalman filter [6]. The ability of estimating nonlinear functions enables the use of EKF in training neural networks. However, the linearization process of EKF involves computing the Jacobian of the network which can be computationally expensive, especially in the case of CSRNs. Further, as the number of cells in CSRN is increased to account for more inputs, the size of Jacobian matrix increases accordingly. Consequently, for large number of network inputs, which is very common for large scale maze and image data, computational complexity of computing Jacobian in EKF training becomes prohibitive [4]. Furthermore, the linear approximation of the nonlinear system model could introduce errors in the estimation process [6] which may adversely affect convergence of the network.

In order to alleviate the drawbacks associated to training CSRN with EKF, this paper introduces UKF, for the first time in literature, to train CSRNs. The UKF implements the unscented transform, which is a method of calculating the statistics of a random variable that undergoes a nonlinear transformation. The UKF uses the true nonlinear system model in the estimation process, and therefore, when used in neural network training, it only requires the NN forward propagation function in parameter estimation. Consequently, one of the main advantages of using UKF to training CSRNs is that the complex Jacobian calculations can be avoided. Also, it is shown in [6] that UKF state estimates are accurate to the second order, while EKF achieves a first order accuracy. However, the computation complexity of the UKF is similar to that of EKF [6].

This work examines the effectiveness of UKF training in CSRN for 2D maze traversal and binary image affine transformation tasks. The results are compared with a CSRN trained with EKF for these applications.

Section II of this paper outlines the 2D maze traversal problem. Section III discusses a brief overview of image affine transforms. Section IV introduces the CSRN and explains the learning algorithms including UKF. The results for CSRN maze traversal, binary image transformations and performance comparison of CSRN with both GMLP and Elman cores along with computational and performance metrics are presented in section V. Finally, section VI provides discussions and conclusions.

II. 2D MAZE TRAVERSAL PROBLEM

2D maze traversal problem consists of finding the optimal path from a starting location of a 2D grid to a certain goal avoiding some obstacles.

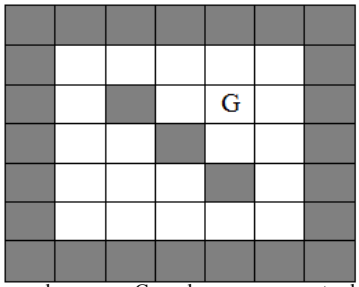


Fig. 1 A 5x5 example maze. Grey boxes represent obstacles and 'G' represents goal. White cells are clear. Walls surrounding the maze make the maze size 7x7

The optimal path can be obtained by computing the *J* cost function using Belman's equation as shown in Eq. (1). An example maze with goal and obstacles is shown in Fig. 1. Each cell of the maze grid is either clear or an obstacle. In order to find the optimal path, the best strategy is to choose the neighboring cell that has the smallest *J* [1] [5].

III. AFFINE TRANSFORMATION OF IMAGE

Affine transformation is a 2D geometric transformation on an image. This is basically a mapping technique where the locations of the intensity values of an input image are mapped to the new location of an output image. Translation, rotation, scaling and shear are the type of affine transformations performed in an image. The affine transformation between two images is given by the following equation,

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x_o \\ y_o \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad (2)$$

where (x_n, y_n) and (x_o, y_o) are the spatial coordinates of the corresponding pixel locations in transformed and original images, respectively. Rotation is performed by the 2x2 rotation matrix with an angle of "θ". "t_x" and "t_y" represent the translation along x and y axis. "S_x" and "S_y" represent the scaling parameter in x and y direction [4] [7]. Detail of affine transformation can be found in [8].

IV. CELLULAR SIMULTANEOUS RECURRENT NETWORK

A. Introduction to CSRN

The CSRN is first introduced by Pang et al. [1]. The authors have shown that this new type of neural network is capable of solving learning problems efficiently when compared to MLP. As an example they have solved 2D maze traversal problem and NetA/NetB problem.

The CSRN is a combination of a cellular network and a simultaneous recurrent network (SRN). Simultaneous recurrent networks are different from standard recurrent networks by the fact that the feed-back from the output is taken without any delay. Theoretically, the inputs and the outputs should be simultaneous. SRNs can mimic the activity of human brain. The core part of a CSRN is SRN.

On the other hand, the cellular network has identical elements in each cell which either can be a single neuron or an entire network. The elements are arranged in some geometric pattern. This kind of structure can be useful to solve problems that have some inherent geometry. A cellular architecture is shown in Fig. 2.

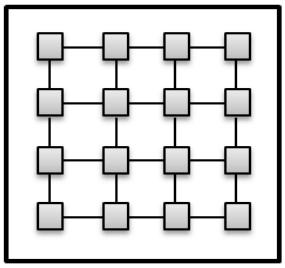


Fig. 2 Cellular architecture

The primary benefit of cellular architecture is weight sharing between different elements. The idea of weight sharing significantly decreases the number of weights, as well as the time needed to train the network.

A CSRN architecture can be constructed by making each cell of the cellular network an SRN. CSRN architecture is shown in Fig. 3. The grey boxes represent the CSRN cell with a SRN core. Each cell of the network receives output from its four neighbors from previous iteration as input. Also each cell provides output to its neighbors.

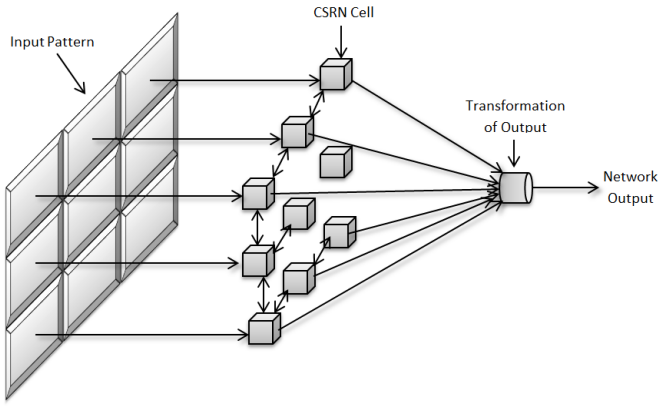


Fig. 3 CSRN architecture

Note that the cellular structure of CSRN matches with the input pattern. Therefore, each cell of the input can be directly fed to each cell of CSRN [9]. The core network is a generalized multi-layered perceptron (GMLP). The detail of GMLP network can be found in [4].

B. Back-Propagation Through Time (BPTT)

Recurrent neural network training can be done using the BPTT algorithm. The BPTT algorithm is an extension of the regular back-propagation algorithm in which the recurrent neural network is 'unfolded' prior to training. Specifically, this 'unfolding' process creates a pseudo feed-forward network consisting of replications of the original network with the recurrent link being fed forward into the successive copy. If the network stabilizes, the output may not change in further replications; in which case the replication process is stopped. The multi-layered feed-forward network resulting from the above process can be considered as equivalent to the recurrent network and this can be trained using the regular back-propagation algorithm. However, the weights in each replication must be equal, and therefore, cannot be updated individually. Weight updating in BPTT is done by updating the weights simultaneously by using the sum of all the derivatives. In the case of cellular SRN, the derivatives must be calculated and summed over each cell of the network [5]. This greatly increases the complexity of back-propagation and affects the training efficiency. BPTT was successfully applied in CSRN training for maze traversal in [10]. However for a single maze, the CSRN learning with BPTT requires around 1000 epochs for convergence.

C. Extended Kalman Filter (EKF)

The Kalman filters, originally proposed by Kalman [11], are commonly used in signal processing applications. Kalman filter essentially provides a computational means to recursively estimate future states of a system based on past observations. The original Kalman filters are linear recursive filters that estimate the state of a linear dynamic system. For estimation of nonlinear models, an extension of Kalman filters referred to as Extended Kalman Filer (EKF) is used. Parameter estimation through Kalman filters have been utilized in neural network training [6]. In this case, the neural network weights are regarded as the parameters to be estimated, and the neural network outputs are regarded as the observations of the system. The basic idea of EKF is to linearize the system model at each iteration prior to applying standard Kalman filter [6]. The linear approximation of the nonlinear model is performed by computing the partial derivative matrices (Jacobian) of the nonlinear state transition and observation functions. In parameter estimation for neural networks, the state transition and observation equations are given as,

$$W_{t+1} = W_t + \gamma_t, \tag{3}$$

and

$$Y_{t+1} = F(W_t, u_t) + \eta_t. \tag{4}$$

In Equation (3), the system state or the neural network weights are denoted by W_t at time t and the process noise is denoted by γ_t. u_t represents the input to the neural network. In the observation equation (Equation 4), Y_{t+1} denotes the observation (NN output) at time $t+1$, and F denotes the forward propagation function of the neural network. η_t represents the measurement noise. Here, for EKF, the Jacobian matrix [6] is calculated using standard back-propagation or BPTT for feed-forward and recurrent networks.

Ilin et al. in [5] successfully trained CSRN with EKF for maze navigation. Anderson et al. [4] used the CSRN trained with EKF for topological image transforms. The authors report a large reduction in training time with EKF. For an example, CSRN trained with EKF for a single maze converges within 15 to 30 epochs while that with BPTT requires approximately 1000 epochs. However, the Jacobian matrix required in EKF parameter estimation for CSRN is calculated using the BPTT algorithm, which can be quite complex due to the structure of the CSRN. Furthermore, the computation of Jacobian in CSRN trained with EKF can be cumbersome as discussed above [4].

D. Unscented Kalman Filter (UKF)

In this work, we propose UKF for the training of CSRN. The UKF was first introduced by Julier et al. [12] and was further developed by van der Merwe et al. [13]. The main difference between EKF and UKF lies in the method of representing random variable for the propagation through a dynamic system [6]. In EKF, the state of the system, approximated by a Gaussian random variable is propagated

through a first order linear estimate of the nonlinear system [6]. This may introduce errors in the transformed mean and covariance of the random variable [6]. In comparison, UKF attempts to solve this problem by first sampling the state approximation and choosing the best sample points that represents the true mean and covariance of Gaussian random variable. These sample points are then propagated through the true nonlinear system [6]. The transformed sample points capture the transformed mean and covariance with a second order accuracy. The UKF utilizes unscented transformation, which is a method of calculating the statistics of a random variable that goes through a nonlinear transformation.

Consider an *n*-dimensional Gaussian random variable, *x* with a mean μ_x, and covariance P_x, that is transformed with by non-linear function *g* as follows,

$$Y = g(x) \tag{5}$$

Instead of computing a linear approximation of *g* as is done in EKF, the unscented transform computes a minimal set of weighted sample points that captures the true mean and covariance of the prior, *x*. These samples are referred to as sigma points. The sigma points, when passed through the true nonlinear function, *g*, captures the mean and covariance of the posterior *y*, with a minimum second order accuracy [14].

A number of $2n + 1$ sigma points must be selected to accurately capture the mean and covariance of the prior, *x*. The sigma points ($\boldsymbol{Z}^{[i]}$) are located at the mean and along the main axes of covariance. The sigma points are calculated as,

$$\boldsymbol{Z}^{[0]} = \mu_x, \tag{6}$$

$$\boldsymbol{Z}^{[i]} = \mu_x + (\sqrt{(n+\lambda).P_x})_i, \qquad for\ i = 1\ to\ n \tag{7}$$

$$\boldsymbol{Z}^{[i]} = \mu_x - (\sqrt{(n+\lambda).P_x})_{i-n}, \qquad for\ i = n+1\ to\ 2n \tag{8}$$

and

$$\lambda = \alpha^2(n+k) - n \tag{9}$$

where *n* is the dimensionality of *x* and $\boldsymbol{Z}^{[i]}$ is the ith sigma point. The parameters α and *k* are scaling parameters that adjusts the spread of sigma points with respect to the mean.

The sigma points are then passed through the nonlinear function as follows,

$$y^{[i]} = g\left(\boldsymbol{Z}^{[i]}\right). \tag{10}$$

The mean and covariance of the posterior *y*, can now be approximated from a weighted sample mean and covariance of the transformed sigma points $y^{[i]}$ as [6],

$$\mu_y = \sum_{i=0}^{2n} w_m^{[i]}.y^{[i]}, \tag{11}$$

and

$$P_y = \sum_{i=0}^{2n} w_c^{[i]}.(y^{[i]} - \mu_y)(y^{[i]} - \mu_y)^T \tag{12}$$

with the weights computed as,

$$w_m^{[0]} = \frac{\lambda}{n+\lambda}, \tag{13}$$

$$w_c^{[0]} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta), \tag{14}$$

and

$$w_m^{[i]} = w_c^{[i]} = \frac{1}{2(n+\lambda)}, \quad for\ i = 1\ to\ 2n \tag{15}$$

where $w_m^{[i]}$ and $w_c^{[i]}$ are the mean and covariance weights of the ith sigma point, and parameter β ($\beta = 2$ is optimal for Gaussian distributions [6]) is chosen to incorporate prior knowledge of the underlying Gaussian.

The UKF implements a Bayesian filter using the unscented transform [15]. When using UKF for training neural networks, the problem is posed as a parameter estimation case, similar to EKF. Here, the NN weights are considered as the state of the system, and the system equations utilized are given as,

$$w_{t+1} = w_t + \varepsilon_t, \tag{16}$$

and

$$y_t = G(w_t, u_t) + \delta_t, \tag{17}$$

where (16) is the state transition and (17) is the measurement equations respectively, ε_t and δ_t are zero mean Gaussian additive process and measurement noise with covariance Q_t and R_t respectively. The nonlinear transformation *G* is the forward propagation function of the neural network and y_t is the resulting output of the neural network.

The current estimate of the system state is given by the mean μ_k^w and its covariance P_k^w. The UKF based NN training starts with the previous estimate of the system state (μ_{k-1}^w, P_{k-1}^w), the current input to the neural network u_k, and the NN target output T_k. The Bayesian filter prediction step is given as,

$$\bar{\mu}_k^w = \mu_{k-1}^w, \tag{18}$$

and

$$\bar{P}_k^w = P_{k-1}^w + Q_{k-1}. \tag{19}$$

The sigma points are then extracted as shown in (6) to (8) to form the vector,

$$\boldsymbol{Z}_k = \left[\bar{\mu}_k^w \qquad \bar{\mu}_k^w + \gamma\sqrt{\bar{P}_k^w} \qquad \bar{\mu}_k^w - \gamma\sqrt{\bar{P}_k^w}\right], \tag{20}$$

where, $\gamma = \sqrt{n+\lambda}$. Next, the sigma points are passed through the non-linear measurement process equation to obtain the measurement update,

$$\boldsymbol{Y}_k^{[i]} = G(\boldsymbol{Z}_k^{[i]}, u_k). \tag{21}$$

The updated state estimate statistics are computed using transformed sigma points $\boldsymbol{Y}_k$ as shown in equations (11) and (12) to obtain,

$$\mu_k^y = \sum_{i=0}^{2n} w_m^{[i]} . \boldsymbol{Y}_k^{[i]}, \tag{22}$$

$$P_k^y = \sum_{i=0}^{2n} w_c^{[i]} . (\boldsymbol{Y}_k^{[i]} - \mu_k^y)(\boldsymbol{Y}_k^{[i]} - \mu_k^y)^T + R_t, \tag{23}$$

and

$$P_k^{w,y} = \sum_{i=0}^{2n} w_c^{[i]} . (\boldsymbol{Y}_k^{[i]} - \bar{\mu}_k^w)(\boldsymbol{Y}_k^{[i]} - \mu_k^y)^T \tag{24}$$

The Kalman gain is computed using above estimates given as,

$$K_k = P_k^{w,y} . (P_k^y)^{-1}. \tag{25}$$

Finally, the system estimation updates are computed for the current state as follows,

$$\mu_k^w = \bar{\mu}_k^w + K_k(T_k - \mu_k^y), \tag{26}$$

and

$$P_k^w = \bar{P}_k^w + K_k . P_k^y . K_k^T. \tag{27}$$

The UKF based training for the CSRN is performed using the same steps by taking CSRN forward propagation function as the nonlinear observation function G in equation (15). The UKF training algorithm for CSRN is shown in Figure 4 [15].

Randomly assign the network weights, w_0 and set $\mu_{k-1}^w = w_0$.
Calculate initial covariance matrix, P_0^w and set $P_{k-1}^w = P_0^w$.
For each epoch
For each training maze/image
Perform Bayesian prediction step given by (18) and (19)
Compute sigma points using (5) - (9)
Compute Measurement updates: CSRN forward-propagation with sigma points (17)
Compute measurement update statistics using sigma point weights (22)-(24)
Compute the Kalman gain (25)
Compute the state estimation update (26), (27)
End
End

Fig. 4 UKF algorithm

Unlike the EKF, the UKF algorithm in Fig. 4 does not need a linear approximation process. UKF uses the true nonlinear observation function for its estimation. Therefore UKF does not have the Jacobian calculation related problems inherent to EKF. Furthermore, as UKF results are accurate to the 2nd order, theoretically it should outperform EKF in weight estimation process.

V. Results and Discussion

A. Maze Navigation

The CSRN is trained using UKF algorithm to solve 2D maze traversal problem. First, a single maze is chosen to train and test the network. The same experiment is repeated with EKF based training. The training is obtained over 200 epochs for both EKF and UKF. Note the number of epochs in this work is fixed to different upper limits by observation. The experiment is performed in an Intel© Core© i7 2GHz machine with 8 GB RAM. All the experiments are performed using this machine. Figure 5(a) shows the target maze, Fig. 5(b) shows the *J* function approximation using UKF and Fig. 5(c) shows the *J* function approximation using EKF respectively. Note Fig. 5 clearly shows that UKF offers better *J* function approximation than EKF.

5	4	3	2	3
6	5	25	1	2
7	6	25	25	3
8	7	6	5	4
9	8	7	6	5

(a)

4.93	4.03	3.01	1.98	2.99
5.99	4.97	25.03	1.01	2.02
7.02	5.97	25.04	25.01	2.99
8.04	6.99	6.03	4.96	4.01
8.98	7.98	7.03	5.99	5.01

(b)

5.43	3.38	3.43	2.14	3.03
5.81	4.67	24.96	0.95	2.63
7.07	5.48	25.59	24.64	2.88
7.65	7.42	5.76	4.51	3.72
8.91	8.45	6.85	6.18	5.12

(c)

Fig. 5 *J* function approximation using CSRN trained by EKF and UKF. (a) Target maze, (b) *J* function approximation with UKF, (c) *J* function approximation with EKF

Figure 6 shows sum squared error (SSE) over the entire 200 epochs for both EKF and UKF. It can be clearly seen that UKF converges faster than EKF. The network converges after 20 epochs when trained by UKF compared to that of 50 epochs by EKF. The final SSE is also smaller for UKF (0.0654) than EKF (8.0611).

Next the CSRN is trained using UKF with 10 randomly generated mazes and then tested with 6 completely different random test mazes. The network is trained with 200 epochs. The purpose of this experiment is to investigate the ability of the CSRN network trained with UKF to learn from a set of random mazes and to apply that knowledge to navigate an unseen maze.

By observing the outputs it can be seen that the network is capable of correctly predicting the optimal path from a starting point to the goal.

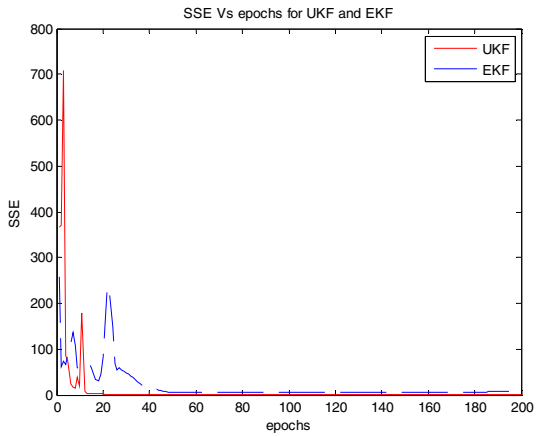


Fig. 6 Sum squared error over epochs for EKF and UKF

Figure 7 (a) and (b) shows one of the testing results and its target maze.

5.25	3.48	24.91	-0.05	1.49
25.05	3.28	1.91	0.73	1.81
7.33	24.96	2.58	2.19	24.93
7.33	6.29	4.02	3.78	6.18
8.07	6.87	25.06	25.05	6.51

(a)

5	4	25	0	1
25	3	2	1	2
7	25	3	2	25
6	5	4	3	4
7	6	25	25	5

(b)

Fig. 7 Testing result, (a) A testing maze, (b) Target maze of (a)

The same experiment is performed using EKF for comparison. Figure 8 shows the testing maze sum squared error (SSE) for EKF and UKF for the entire 200 epochs. Here again UKF offers significant improvement over EKF. The detailed result of this specific experiment with multiple random mazes for CSRN trained with EKF can be found in [2].

Note in Fig. 8, the UKF testing appears somewhat unstable representing random fluctuations in the error curve. This may be due to local minima in UKF optimization process; however, this needs further investigation. Furthermore, the overall error in UKF is still much less than that for EKF.

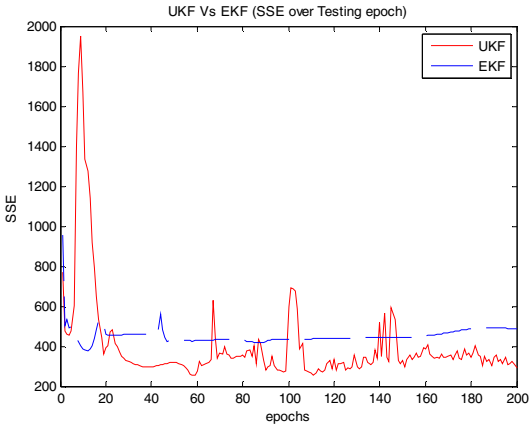


Fig. 8 Comparison of EKF and UKF in terms of SSE (Testing)

The processing time for UKF and EKF to train the network with a single maze and with ten mazes is shown in Table I.

TABLE I
TRAINING TIME OF UKF AND EKF

	Processing Time (Seconds)	
	UKF	EKF
Training one maze	16.734	1.81671
Training ten mazes	162.93	21.22

As expected, Table I shows that UKF training is slower than EKF. The higher processing time in UKF is due to the fact that each sigma points generated in the unscented transform process has to propagate through the nonlinear observation function (CSRN forward propagation) sequentially. Since the number of sigma points generated is related to the dimensionality of the state random variable (CSRN network weights), having a relatively large number of weights in CSRN increases the processing time. This shows that even though UKF training is slower than EKF, it produces better convergence. This suggests that CSRN trained with UKF is capable to approximate *J* function for ADP better than that trained with EKF.

B. Binary Image Transformation

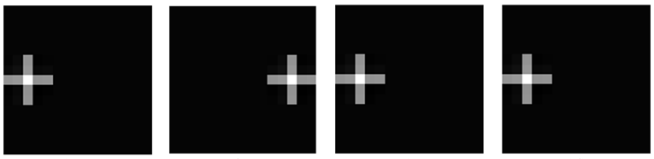
The CSRN trained with EKF has been successfully implemented for binary image affine transformation [4]. In this section CSRN trained with UKF is evaluated for binary image transformations. Several key image statistics introduced in [4] have been used in this paper to evaluate the performance of binary image transformation. Brief description of these metrics is given below [4].

- J_{acc} - function accuracy. Each network cell output is compared with the corresponding transformation function value for that cell. When the values are equal it is considered as a match. Total number of match is then normalized by the number of cells to get the accuracy.
- IM_{acc} - image accuracy. Pixel by pixel comparison of the output image and the target image. If the pixel values are same, it is considered as a match. The ratio of the number of matched pixels and the total number of pixels gives the percentage image accuracy.
- IM_{cr} - Normalized cross correlation between output image and target image. Correlation value closer to 1 means better similarity between the images.

Several binary images of size 15x15 pixels is used for evaluation and comparison. A simple binary cross image is used for the experiments. The results of binary image transformations are given below.

1) Translation

For the translation task, a binary cross image of size 5x5 overlayed on a 15x15 blank image is used. The CSRN is trained with UKF using 10 training images each moved by one pixel in x direction and trained with 50 epochs. The network is then tested with an image translated by 10 pixels. The same experiment is performed by CSRN trained with EKF for comparison. Figure 9(a) shows the target image, Fig. 9(b) shows the input image, Fig. 9(c) shows the CSRN output trained with UKF and Fig. 9(d) shows CSRN output trained with EKF, respectively.



(a) (b) (c) (d)

Fig. 9 CSRN image translation results, (a) Target image, (b) Input image, (c) Output of CSRN trained with UKF, (d) Output of CSRN trained with EKF. For UKF and EKF the network achieves IM_{acc} = 100%, J_{acc} = 86.67% for UKF and 60% for EKF.

In Fig. 9, for both UKF and EKF the network achieves 100% image accuracy in translating the test image. However, UKF performs better in approximating the transformation function values. Table II summarizes the binary translation results for CSRN trained with UKF and EKF respectively. Similar to maze traversal case, CSRN trained with UKF requires more time when compared to that with EKF.

SSE of translation task over 50 testing epochs for both UKF and EKF is shown in Fig. 10. Here again the fluctuations in the UKF error curve can be observed as discussed above.

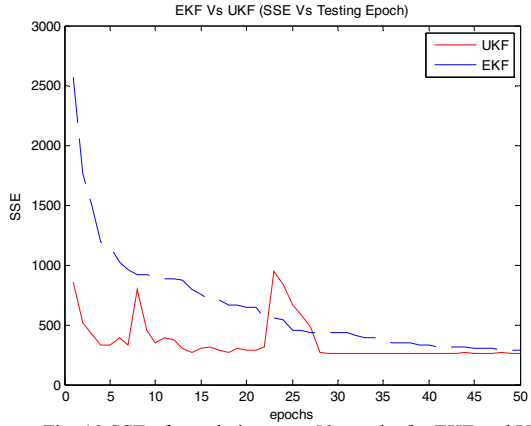
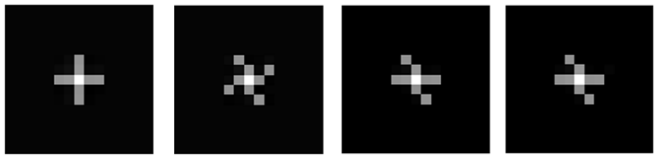


Fig. 10 SSE of translation over 50 epochs for EKF and UKF

2) Rotation

Next UKF trained CSRN is applied for rotation transformation. Rotation is a more complex transformation compared to translation in which each pixel in an image is subjected to different transformation values. Therefore, rotation evaluates the ability of CSRN trained with UKF to approximate complex transformation functions. For the rotation case, the same reference image in translation experiment is used to generate the training image set. The CSRN is trained with 11 training images each rotated by 2^0 that ranges from 0^0 to 20^0. The network is then tested with an image rotated by 16 degrees. The test is also conducted with EKF trained CSRN for comparison. The rotation results are shown in Fig. 11. Note the number of training images and all other values are set by observation in this work.

Although the result of UKF and EKF in Fig. 11 appears similar, UKF offers slightly better image accuracy when compared to EKF. Further, the function approximation accuracy is higher in UKF.

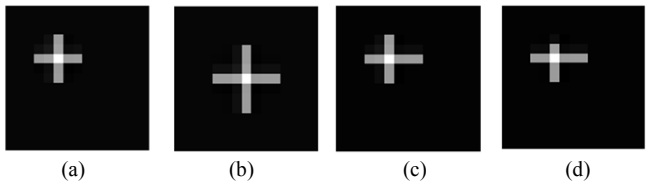


(a) (b) (c) (d)

Fig. 11 CSRN image rotation results, (a) Target image, (b) Input image, (c) Output of CSRN trained with UKF, (d) Output of CSRN trained with EKF. For UKF and EKF the network achieves IM_{acc} = 94.22% & 93.33%, J_{acc} = 61.33% & 52.00% for UKF and EKF respectively

3) Scaling

Finally Scaling is performed using CSRN trained with UKF and EKF. Scaling is also a complex transformation operation similar to rotation. The experiment is performed on down scale operation, for an example. The CSRN is trained for 50 epochs with four training images generated from a reference image of size 7x7 binary cross overlayed on a 15x15 blank background image. The four training images are constructed such that each image is a 85% scaled down version of the previous image to produce 62%, 73%, 86% and 100% of the original image size respectively. The testing is performed on the 73% scaled version. Figure 12 shows the scaling results.



(a) (b) (c) (d)

Fig. 12 CSRN image scaling results, (a) Target image, (b) Input image, (c) Output of CSRN trained with UKF, (d) Output of CSRN trained with EKF. For UKF and EKF the network achieves IM_{acc} = 94.67% & 92.44%, J_{acc} = 35.56% & 30.22% for UKF and EKF respectively

Table II summarizes all the affine transformation results.

TABLE II
AFFINE TRANSFORMATION RESULT

Metric	Translation		Rotation		Scaling	
	UKF	EKF	UKF	EKF	UKF	EKF
J_{acc} (%)	86.67	60.00	61.33	52.00	35.56	30.22
IM_{acc} (%)	100.00	100.00	94.22	93.33	94.67	92.44
IM_{cr}	1.00	1.00	0.82	0.82	0.96	0.91
Training Time (Sec.)	104.4	29.4	112.2	31.2	60.6	11.4

Table II shows that UKF shows comparable results for transformed image accuracy metric, IM_{acc}, and better transformation function approximation accuracy metric J_{acc} when compared with EKF.

In order to compare the CSRN network performance with a different core architecture, an Elman recurrent core is implemented. Unlike the GMLP core used, the Elman core is designed with three layers, an input layer, a hidden layer and an output layer. The comparison of performance between the CSRN with GMLP core and the CSRN with Elman core trained with UKF is shown in Table III. The comparison is

shown using the metric, image accuracies (IM_{acc}), that represents the difference between target and output images.

TABLE III
PERFORMANCE COMPARISON BETWEEN CSRN WITH GMLP CORE VS. ELMAN CORE TRAINED WITH UKF

Transformation	CSRN with GMLP Core	CSRN with ELMAN Core
Translation (%)	100	100
Rotation (%)	94.22	95.56
Scaling (%)	94.67	94.67

Table III shows that the performance of CSRN with GMLP core is comparable to that of CSRN with Elman core.

The results in Tables II show that the processing time for UKF is higher than EKF as explained above. However, note also that UKF does not require computation of Jacobian matrix unlike EKF. Therefore, UKF training is expected to offer faster training involving larger input array with more complex CSRN structures with that of EKF. To investigate this hypothesis, an experiment is conducted on binary image translation using images of different sizes. The CSRN is trained with both UKF and EKF over 10 epochs for each image size. Other parameters are set similar to the previous binary image translation experiment. The processing time for different image sizes are shown in Table IV.

TABLE IV
PROCESSING TIME OF UKF AND EKF FOR DIFFERENT IMAGE SIZE

Image Size	Processing Time (Minutes)	
	UKF	EKF
15x15	0.32	0.16
25x25	0.87	0.86
35x35	1.89	3.57
45x45	3.72	12.06

Table IV shows that computation using UKF is faster as image size increases. For a 15x15 image the processing time of UKF is twice than that of EKF. However, for an image of size 45x45 the processing time of UKF is one fourth of that of EKF. Therefore, we conclude that the CSRN trained with UKF is more efficient in solving more complex problems such as large scale image processing and maze traversal when compared to that of EKF.

VI. CONCLUSION

In this work, we introduce UKF as a learning algorithm for CSRN. We have investigated the performance of a CSRN trained with UKF for solving two representative problems such as 2D maze navigation and binary image affine transformations. We show that UKF can successfully train CSRN to perform both tasks. We systematically compare the performance of CSRN trained with UKF against that of EKF and observe that UKF generally converges faster than EKF. For 2D maze traversal problem, CSRN trained with UKF performs better than that of EKF in terms of both accuracy and convergence. For binary image affine transformation problem, UKF performance is comparable to EKF. We also show that for large scale complex data processing, UKF training is faster than that of EKF. This property is desirable for large image processing tasks. We further compare the performance of CSRN with GMLP core to that of CSRN with an Elman core. The results show that CSRN with both these core architectures trained with UKF are capable of performing binary image transformations successfully. Our future aims include introduction of UKF training in CSRN for more complex image processing tasks such as grey-scale image processing, face recognition, and pattern recognition.

REFERENCES

[1] X. Pang and P. Werbos, "Neural Network Design for J Function Approximation in Dynamic Programming," vol. 1, ed: arXiv:adap-org/9806001, June 1998.

[2] K. Tae-Hyung and D. C. Wunsch, "Modified cellular simultaneous recurrent networks with cellular particle swarm optimization," IEEE World Congress on Computational Intelligence (WCCI), Brisbane, Australlia, June 10-15, 2012, pp. 1-8.

[3] L. L. Grant and G. K. Venayagamoorthy, "Voltage prediction using a Cellular Network," *Power and Energy Society General Meeting, 2010 IEEE*, 2010, pp. 1-7.

[4] J. K. Anderson and K. M. Iftekharuddin, "Learning topological image transforms using cellular simultaneous recurrent networks," *Neural Networks (IJCNN), The 2013 International Joint Conference on*, 2013, pp. 1-9.

[5] R. Ilin, R. Kozma, and P. J. Werbos, "Cellular SRN Trained by Extended Kalman Filter Shows Promise for ADP,"*Neural Networks, 2006. IJCNN '06. International Joint Conference on*, 2006, pp. 506-510.

[6] S. Haykin, "*Kalman Filtering and Neural Networks*." New York: Wiley, 2001, pp. 221-245.

[7] A. B. Abche, F. Yaacoub, A. Maalouf, and E. Karam, "Image Registration based on Neural Network and Fourier Transform," *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, 2006, pp. 4803-4806.

[8] A. K. Jain, *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall, 1989.

[9] T. Salan and K. M. Iftekharuddin, "Large pose invariant face recognition using feature-based recurrent neural network," *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 2012, pp. 1-7.

[10] P. J. Werbos and P. Xiaozhong, "Generalized maze navigation: SRN critics solve what feedforward or Hebbian nets cannot," *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, 1996, vol.3, pp. 1764-1769.

[11] R. E. Kalman and R. S. Bucy, "New Results in Linear Filtering and Prediction Theory," *Journal of Basic Engineering,* vol. 83, 1961, pp. 95-108.

[12] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," *American Control Conference, Proceedings of the 1995*, vol.3, 1995, pp. 1628-1632

[13] E. A. Wan and R. Van der Merwe, "The unscented Kalman filter for nonlinear estimation," *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, 2000, pp. 153-158.

[14] R. van der Merwe, E. Wan, and S. Julier, "Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor-Fusion: Applications to Integrated Navigation," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2004, pp. 1735-1764.

[15] J. K. Anderson, "Cellular Simultaneous Recurrent Networks for Image Processing," Electrical Engineering, Department of Electrical and Computer Engineering, University of Memphis, July 2013.