Person Movement Prediction Using Neural Networks

Lucian Vintan[1], Arpad Gellert[1], Jan Petzold[2], and Theo Ungerer[2]

[1]Computer Science Department, University "Lucian Blaga",
E. Cioran Str., No. 4, Sibiu-550025, Romania
{lucian.vintan, arpad.gellert}@ulbsibiu.ro

[2]Institute of Computer Science, University of Augsburg,
Eichleitnerstr. 30, 86159 Augsburg, Germany
{petzold, ungerer}@infomatik.uni-augsburg.de

Abstract. Ubiquitous systems use context information to adapt appliance behavior to human needs. Even more convenience is reached if the appliance foresees the user's desires and acts proactively. This paper proposes neural prediction techniques to anticipate a person's next movement. We focus on neural predictors (multi-layer perceptron with back-propagation learning) with and without pre-training. The optimal configuration of the neural network is determined by evaluating movement sequences of real persons within an office building. The simulation results, obtained with one of the pre-trained neural predictors, show accuracy in next location prediction reaching up to 92%.

1 Introduction

Ubiquitous systems strive for adaptation to user needs by utilizing information about the current context in which a user's appliance works. A new quality of ubiquitous systems may be reached if context awareness is enhanced by predictions of future contexts based on current and previous context information. Such a prediction enables the system to proactively initiate actions that enhance the convenience of the user or that lead to an improved overall system.

Humans typically act in a certain habitual pattern, however, they sometimes interrupt their behavior pattern and they sometimes completely change the pattern. Our aim is to relieve people of actions that are done habitually without determining a person's action. The system should learn habits automatically and reverse assumptions if a habit changes. The predictor information should therefore be based on previous behavior patterns and applied to speculate on the future behavior of a person. If the speculation fails, the failing must be recognized, and the predictor must be updated to improve future prediction accuracy.

For our application domain we chose next location prediction instead of general context prediction. The algorithms are also applicable for other more general context domains.

This paper focuses on a neural prediction approach, introducing the local and global neural predictors and comparing the neural predictors with and without pre-

training. Our application predicts the next room based on the history of rooms, visited by a certain person moving within an office building. The prediction information is used in the Smart Doorplate application at the University of Augsburg [11] to notify a visitor about the potential return of an absent office owner. We evaluate these neural predictors by some movement sequences of real persons of the research group at the University of Augsburg [8]. The next sections describe the related work, the proposed neural network, and the simulation results.

2 Related Work

To predict or anticipate a future situation learning techniques as e.g. Markov Chains, Bayesian Networks, Time Series or Neural Networks are obvious candidates. The challenge is to transfer these algorithms to work with context information.

Mozer [6] proposed an Adaptive Control of Home Environments (ACHE). ACHE monitors the environment, observes the actions taken by the inhabitants, and attempts to predict their next actions, by learning the anticipation needs. Mozer [7] uses as a predictor a feed-forward neural network with one hidden layer for anticipating the next action (as an example, the system will predict when an inhabitant returns home and therefore will start the heater).

Aguilar et al. [1] implemented a system to reduce latency in virtual environment applications, where virtual images must be continuously stabilized in space against the user's head motion in a head-mounted display. Latencies in head-motion compensation cause virtual objects to swim around instead of being stable in space. To address this problem, Aguilar et. al. used machine learning techniques to define a forward model of head movement based on angular velocity information. They use a recurrent neural network to capture temporal patterns of pitch and yaw motion. Their results demonstrate an ability of the neural network to predict head motion up to 40 ms ahead thus eliminating the main source of latencies.

Otherwise neural network approaches are often used in ubiquitous systems for context recognition (see e. g. [4]), not for context prediction.

Petzold et al. [9] transformed some prediction algorithms used in branch prediction techniques of current high-performance microprocessors to handle context prediction. They proposed various context prediction techniques based on previous behavior patterns, in order to anticipate a person's next movement. The evaluation was performed by simulating the predictors with behavior patterns of people walking through a building as workload. Their simulation results show that the context predictors perform well but exhibit differences in training and retraining speed and in their ability to learn complex patterns. In [10] Petzold et al. compared these predictors with the Prediction by Partial Matching (PPM) method, and they evaluated the predictors by movement sequences of real persons within an office building reaching up to 59% accuracy in next location prediction without pre-training and, respectively, up to 98% with pre-training.

3 The Neural Prediction Approach

The artificial neural networks (NN) are composed of a multitude of neurons representing simple processing elements that operate in parallel [3]. A great advantage of the artificial neural networks is their capacity to learn based on examples (supervised learning). In order to solve a problem traditionally, we have to elaborate its model, and after that we have to indicate a succession of operations that represents the solving algorithm of the problem. However there are practical problems with a high level of complexity, and for this kind of problems it is very hard or even impossible to establish a deterministic algorithm.

In the connectionist models like neural networks we are not forced to give a solving algorithm dedicated to a certain problem; we have to offer to the NN only a multitude of consistent examples in order to learn and generalize them. The network extracts the information from the training samples. In this way it is able to synthesize implicitly a certain model of the problem. In other words, the neural network builds up alone an algorithm to solve a problem. The capacity of the neural network to solve complex practical problems using a multitude of samples gives them a highly large potential of applicability.

3.1 The neural network's structure

We chose a multi-layer perceptron with one hidden layer (see Fig. 1) and back-propagation learning algorithm. The rooms and the persons are binary codified to save computing cost, which is of particular interest for mobile (energy restrictions) or fast moving (real-time restrictions) applications. Thus we chose bit encoding with complexity log_2N (entries in NN), instead of one-room-one-neuron encoding with complexity N (entries in NN). This codification might be useful taking into account further enlargements of the project, too (N will probably grow).

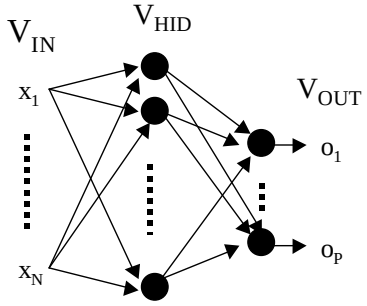

Fig. 1. The multi-layer perceptron

We analyzed two predictor types: the local and respectively the global predictors. In the case of local predictors, each person has his/her own neural predictor and in this way each neural network will be trained with the movements of a single person.

Alternatively, we use one global neural network for all persons, and in this second case the persons must be codified, too.

The input layer: If we use a global predictor the network's input data consists of two codes: the code of the person and the code of the last rooms visited by that person. If we treat each person separately with his/her own predictor, the input data only consists of the codes of the last visited rooms. For detailed information about the codification see [12]. One of the parameters of the network is the number of rooms in the input vector. We'll vary this parameter between 1 and 6, in order to see how the prediction accuracy is affected by the length of room history.

As an example, at a certain moment, for a history of four rooms we would have the following input vectors obtained after a binary codification of the input data:

1. The input vector for a local predictor (for a maximum of 16 rooms, a binary codification of 4 bits is enough): V_{in} = `0101 0010 0001 0011`
2. The input vector for the global predictor (more rooms than 16 must be regarded, using a 5 bit room codification): V_{in} = `01 00101 00010 00001 00011` where the person's code is `01`.

The hidden layer: We will vary the number of neurons in the hidden layer (M cells). We will try first N, $N+1$, $N+2$ (where N is the number of neurons in the input layer) because this was the best configuration of a neural network used in prior work [2].

The output layer: The neural network will return through its output layer the predicted room codified with 4 bits by the local predictor and respectively with 5 bits by the global predictor. In other words, in this concrete case, the output layer of a local neural network will have four neurons ($P = 4$), and the output layer of a global neural network will have five neurons ($P = 5$). As an example, if the binary code of the predicted room is 2 the network will return the following output vector:

1. V_{out} = `0010` - returned by the local predictor
2. V_{out} = `00010` - returned by the global predictor

The neural network's training: For the training/learning process we used the well-known Back-Propagation Algorithm [5], adapted as below:

1. Create a feed-forward network with N inputs, $M = N,\ N+1,\ N+2$ hidden units and P output units.
2. Initialize all network weights $W^1_{i,j}$; $i = \overline{1, N}$; $j = \overline{1, M}$ and $W^2_{i,j}$; $i = \overline{1, M}$; $j = \overline{1, P}$, to small random numbers belonging to the $\left[-\frac{2}{N}, \frac{2}{N}\right]$ interval.
3. Until $E(\overline{W}) = \frac{1}{2} \sum_{k \in Outputs(P)} (t_k - O_k)^2 \le T$ (threshold), do:

   3.1. Input the instance $\overline{X}$ to the network and compute the output $\overline{O}$.

$$\overline{O} = \overline{X} \cdot \overline{W^1} \cdot \overline{W^2} \qquad (1)$$

3.2. For each network output unit k, $k = \overline{1,P}$, calculate its error term δ_k.

$$\delta_k = O_k(1 - O_k)(t_k - O_k) \quad (2)$$

3.3. For each hidden unit h, $h = \overline{1,M}$, calculate its error term δ_h.

$$\delta_h = O_h(1 - O_h) \sum_{k \in Outputs(P)} W_{k,h}^2 \cdot \delta_k \quad (3)$$

3.4. Update each network weight $W_{i,j}$

$$W_{i,j} = W_{i,j} + \Delta W_{i,j} \quad (4)$$

$$\Delta W_{i,j} = \alpha \cdot \delta_i \cdot X_{i,j} \quad (5)$$

where α is the learning step.

The weights will be randomly initialized in the $\left[-\frac{2}{N}, \frac{2}{N}\right]$ interval, where N is the number of neurons in the input layer. For better results we will codify the input data with -1 and 1 and we'll use the following activation function:

$$F(x) = \frac{2}{1+e^{-x}} - 1 \quad (6)$$

Static Learning (Pre-training): The static learning means that the predictor will be trained based on some room history patterns belonging to the previous presented benchmarks (person room movements) before effective run-time prediction process. A very important parameter is the threshold T. As an example, for a threshold of 0.2, the output values are accepted only if they belong to the [-1, -0.8] interval for –1 (0) or in the [0.8, 1] interval for 1. If the output values are not in one of those intervals, the backward step is generated until this condition is fulfilled. In other words, this training iterative process will continue until the error function will be less then the threshold T (0.2 in this case). Another important parameter is the learning rate α.

The static learning process for a local predictor consists in training the network using the person's recorded movements. We'll measure the accuracy gain generated by this static training process. The static learning process for a global predictor consists in alternatively (round robin) training the predictor using input vectors belonging to each benchmark in a supervised manner, using back-propagation algorithm. So, we expect to avoid the undesired forgetfulness process during the training.

Dynamic Learning (Run-time Prediction Process): One of the differences between the static and dynamic learning is that during the dynamic learning we predict based on the feed-forward step's result. That means that if the output value is belonging to [-1, 0) interval it will be considered –1 (0) and if it belongs to the [0, 1] interval it will be considered 1.

If the predicted value is correct only a backward step is made. If the predicted value is not correct, the backward step will be applied until the prediction is correct and one more time after that. This solution could generate some real-time problems. An other more realistic and more attractive solution for PDAs, is to apply the backward step only one time even if the prediction is not correct (this means that the prediction process is faster, and, thus, better adapted to real-time restrictions).

Static & Dynamic or only Dynamic Learning: In the case of static & dynamic learning the network is statically trained before its effective use. In this case the dynamic learning process is started with the weights generated by the static training process. If we use a global predictor the neural network will "learn" randomly the benchmarks during the effective run-time prediction process, too (dynamic learning process). An iteration step means to run one time all the benchmarks. During each iteration step we select randomly the running sequence of the benchmarks. We will study how the number of iterations will affect the prediction accuracy. If we use only dynamic learning the weights are initially randomly generated, and, after this, the network will effectively predict.

4 The simulator and the steps of the simulation. Experimental Results

The developed simulator exhibits the following parameters: the number of neurons in the input layer (N) practically determined by the room history length, the number of neurons in the hidden layer (M), the threshold's value used in the static learning process (T), and the learning rate (α). The simulator's output represents the predicted room. We will vary all these parameters, obtaining in this way the optimal configuration of the neural network. We used two benchmark types reporting the movements of three employees and the boss: the movement sequences reported during the summer 2003 contain about 100-450 movements per person and those of the fall 2003 contain about 1000 movements per person. Our evaluations are based on the fall benchmarks. In case of pre-training we use the summer benchmarks for training. These benchmarks are compliant to the Augsburg Indoor Location Tracking Benchmarks [8].

We begin varying the number of neurons in the hidden layer, and we start with a history length of 2 rooms and a learning rate of 0.3. We try to find a formula for determining the optimal number of neurons' in the hidden layer as a function of the neurons' number in the input layer. After we establish the optimal solution for the neurons' number in the hidden layer, we continue our simulations varying the number of backward steps corresponding to run-time prediction process, and, after that, the learning rate. More important, after we fix all these parameters, we study how the prediction accuracy is affected by the room's history length (and implicitly by the number of neurons in the input layer). Another goal is to study after how many iterations the prediction accuracy will be established (constant). We determine the optimal threshold value for a statically trained dynamic room predictor. We determine the best learning type (static & dynamic or only dynamic), and for doing this, we

compare the prediction accuracy of an only dynamically trained predictor with the accuracy of a statically and dynamically trained predictor using the same simulation parameters. We finish our study comparing the global neural predictor's accuracy with the local predictor's accuracy.

The first parameter we vary is the number of neurons in the hidden layer. For this we used a dynamically trained network with a learning rate of 0.3 and a room history length of 2. Another goal is to determine after how many iterations the prediction accuracy will be saturated. We consider that the prediction accuracy is saturated if the difference between the prediction accuracies obtained in the last two iterations is less than 0.01. We use the fall benchmarks and two predictor types, the local predictor and the global one (see section 3). Table 1 shows how the prediction accuracy is affected by the number of neurons in the hidden layer.

Table 1. Study of the number of neurons in the hidden layer (M); AM=Arithmetic Mean

Predictor	M=5	M=7	M=9	M=11	M=13	M=15
Employee 1	74.16	75.93	76.01	75.95	75.83	75.6
Employee 2	70.65	71.22	71.38	71.26	71.13	71.04
Employee 3	68.27	68.71	69.43	69.4	69.28	69.18
Boss	58.69	60.07	60.47	59.53	58.69	56.96
AM	67.94	68.98	69.32	69.03	68.73	68.19
Global	56.78	56.96	59.62			

Table 2. The number of iterations needed for saturated prediction accuracy

Predictor	M=5	M=7	M=9	M=11	M=13	M=15
Employee 1	14	36	34	36	33	36
Employee 2	34	29	31	28	17	17
Employee 3	17	9	20	17	11	18
Boss	28	15	26	11	11	8
Global	10	10	16			

As we can see the optimal number of hidden layer neurons is 9, in the case of the local predictors. If we want a formula for the calculation of neurons' number in the hidden layer as a function of the neurons' number in the input layer, we could consider that the optimal number of hidden layer neurons M = N+1, whereas N is number of input layer neurons, because our local predictors, for a room history length of 2, have 8 neurons in the input layer (see section 3).

We continue our study varying the maximum number of backward steps. We used for that the fall benchmarks and a dynamically trained predictor with N+1 hidden layer neurons, a learning rate of 0.3 and a room history length of 2. We also limited the number of iterations to 10. Table 3 shows how the prediction accuracy is affected by the number of backward steps.

As we can see the optimal number of backward steps is 1. We continued our simulations using only one backward step in the dynamic learning process.

Table 3. Study of the number of backward steps (NB)

Predictor	NB=1	NB=2	NB=3	NB=4	NB=5	NB=unlimited
Employee 1	75.58	75.44	75.97	75.62	75.34	74.9
Employee 2	74.04	72.03	71.61	70.86	71.12	70.32
Employee 3	70.19	69.93	69.25	69.49	68.22	68.57
Boss	70.29	65.94	64.32	64.16	63.14	58.72
AM	72.525	70.835	70.2875	70.0325	69.455	68.1275
Global	63.35	61.57	60.21	59.85	60.73	

The next varied parameter is the learning rate. We used the fall benchmarks and a dynamically trained predictor with (N+1) hidden layer neurons and a room history length of 2 and we limited the number of iterations to 10. Table 4 shows how the prediction accuracy is affected by the learning rate. We chose the learning rate of 0.1 in the next simulations (some learning rates in Table 4 given even better result, e.g. 0.05 for the predictor of the boss or 0.15 for the global predictor).

Table 4. Study of the number of learning rate (α)

Predictor	α =0.05	α =0.1	α =0.15	α =0.2	α =0.25	α =0.3
Employee 1	74.55	75.11	75.18	75.72	75.76	75.58
Employee 2	74.68	74.67	74.16	74.08	73.99	74.04
Employee 3	71.44	70.75	70.25	70.31	70.75	70.19
Boss	70.37	70.08	70.05	70.24	70.13	70.29
AM	72.76	72.6525	72.41	72.5875	72.6575	72.525
Global	69.58	70.18	70.32	69.58	65.95	63.35

We continue by the room history length variation, using a dynamically trained predictor with (N+1) hidden layer neurons, a learning rate of 0.1, a single backward step, and also we limited the number of iterations to 10. For these simulations we used the fall benchmarks. Table 5 and Fig. 2 show how the prediction accuracy is affected by the room history length. We can observe that the optimal room history length is 2. If we increase the rooms' history, the prediction accuracy decreases.

Table 5. Study of the room history length (H)

Predictor	H=1	H=2	H=3	H=4	H=5	H=6
Employee 1	71.29	75.11	74.87	74.76	74.63	74.08
Employee 2	65.86	74.67	73.73	73.27	73.08	72.63
Employee 3	61.06	70.75	70.12	70.59	70.38	69.4
Boss	66.7	70.08	70.96	70.08	68.62	68.32
AM	66.2275	72.6525	72.42	72.175	71.6775	71.1075
Global	62.57	70.18	68.59	68.98	66.6	64.32

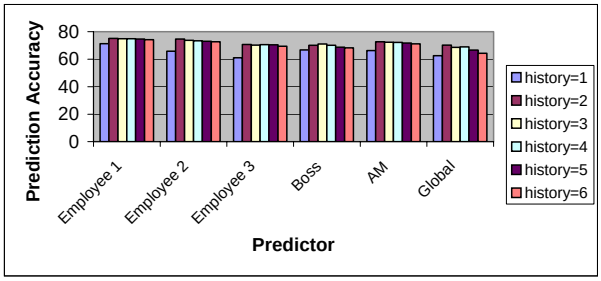

Fig. 2. Study of the room history length

We continue our study implementing a statically trained neural network. The last parameter we varied is the threshold used in the static training process. For this we use a dynamic neural predictor, which was statically learned before it is effective use. That means that the dynamically trained predictor is initialized with the weights generated by the static learning process. We used N+1 hidden layer neurons, a room history length of 2, a learning rate of 0.1, a single backward step in the dynamic training process, and also we limited the number of iterations to 10. For these simulations we used the summer benchmarks in the static training process and respectively the fall benchmarks in the dynamic learning process. Table 6 and Fig. 3 show how the prediction accuracy is affected by the threshold's value used in the static training process.

Table 6. Study of the threshold in the static training process (T)

Predictor	T=0.1	T=0.3	T=0.5	T=0.7	T=0.9
Employee 1	76.44	76.34	76.27	76.18	76.13
Employee 2	75.28	75.04	75.03	75.04	74.98
Employee 3	71.87	71.82	71.51	71.48	71.43
Boss	72.32	72.18	72.16	72.18	71.97
AM	73.9775	73.845	73.7425	73.72	73.6275
Global	72.11	71.74	71.53	71.47	71.32

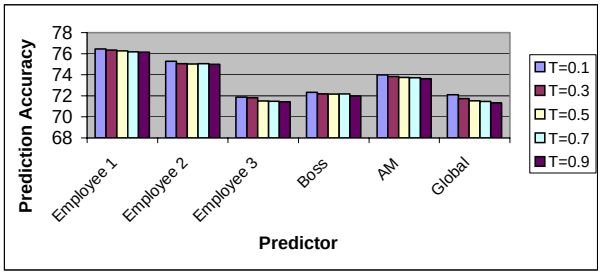

Fig. 3. Study of the threshold in the static training process (T)

The optimal threshold is 0.1, and if we increase it then the prediction accuracy decreases.

We extracted from the presented previous results the prediction accuracies obtained when the prediction process is simplified. In this case we'll be interested only in predicting that the person will return in his own room. Obviously, the prediction is generated only if that person is not in his/her own room. We compared the predictors with and without pre-training using (N+1) hidden layer neurons, a room history length of 2, a learning rate of 0.1, a single backward step in the dynamic training process, and also we limited the iterations' number to 10. For the static training process we used a threshold of 0.1, too. In the static training process we used the summer benchmarks and in the run-time prediction process were used the fall benchmarks.

Table 7. Comparing a dynamic predictor with a statically trained dynamic predictor

Predictor	Dynamic training	Static & Dynamic training
Employee 1	89.32	92.32
Employee 2	89.55	91.21
Employee 3	85.89	87.66
Boss	84.49	88.06
AM	87.31	89.81
Global	84.58	87.3

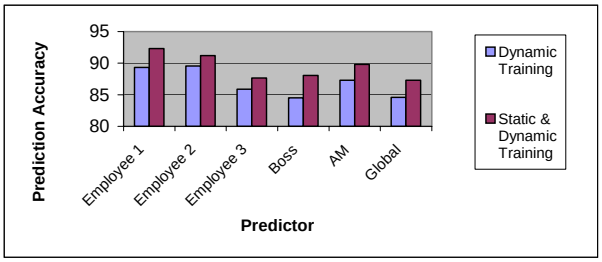

Fig. 4. Comparing a dynamic predictor with a statically trained dynamic predictor

As we can see the best results were obtained with the statically trained dynamic predictor. Also the best results were obtained when we used the local predictors (person – centric).

Time and memory costs: The costs of the approach (time and memory size) are the following:

- Time costs: For static learning the neural network needs about 10 to 45 seconds to learn an entire summer benchmark, using a Pentium III, 650 MHz, 128 MB RAM. The dynamic learning is practically instantaneous because we use a single backward step.
- Memory costs: For a local predictor with a room history length of 2 (H=2), codifying the room with 4 bits (B=4), we have N=B*H=8, M=B*H+1=9, P=B (N/M/P - the number of input/hidden/output layer neurons). For this optimal configuration of the neural network, the system needs 168 memory units (160 float

value memory units, and 8 bits for the input vector). More generally, the memory costs (C) are given by the following formula:

$C = M(N+B) + P(M+B) + N$	- the number of memory units
$C_F = M(N+B)+P(M+B)$	- float value memory units
$C_B = N$	- the number of memory units necessary to store 1 or -1 (1 bit)

5 Conclusions

This paper analyzed neural prediction techniques used in an ubiquitous computing application. In ubiquitous environments often relatively simple prediction algorithms are required e.g. due to the PDA's memory, computing, and communication restrictions. We used in this work one of the simplest neural networks, a multi-layer perceptron with one hidden layer, trained with back-propagation algorithm.

Two predictor types were analyzed: the local and respectively the global predictors. In the case of local predictors, each person has his/her own neural predictor and in this way each neural network will be trained with the movements of a single person. Alternatively, it is possible to use one global neural network for all persons, and in this second case the persons must be codified, too. The evaluations show that the local predictors have higher prediction accuracy than the global predictors, probably due to benchmarks interferences. Therefore, the global predictor was not able to exploit potential useful correlations between different persons' movements. We found a formula for determining the optimal number of neurons in the hidden layer as a function of the neurons' number in the input layer; we could consider that the optimal number of hidden layer neurons $M = N+1$, whereas N is number of input layer neurons. The neural network is more efficient when only one backward step is applied in the run-time prediction process. The next varied parameter was the learning rate. The evaluations show that the learning rate of 0.1 is near to optimal. More important, after we fixed all these parameters, we studied how the prediction accuracy is affected by the room history length (and implicitly by the number of neurons in the input layer). The simulation results show that the optimal room history length is 2. We continued our study implementing a statically trained neural network. The last parameter we varied is the threshold used in the static training process. For this we used a dynamic neural predictor, which was statically learned before it's effective use. The results show that the optimal threshold is 0.1.

We extracted from the presented previous results the prediction accuracy obtained using a simplified prediction process. We compared the dynamic predictor with the statically trained dynamic predictor. The experimental results show that the pre-trained dynamic predictors are more efficient than the dynamic predictors. The arithmetical mean of the prediction accuracies obtained with the pre-trained local predictors is 89.81%, but the prediction accuracy measured on some local predictors grew up to over than 92%. For an efficient evaluation, the static training process was made with some summer benchmarks, and in the run-time prediction process were used fall benchmarks. One of the further development directions is to compare the

neural predictors presented in this work with other neural predictors (which might exploit some inter-benchmarks correlations) and with the state predictor techniques (proposed in [9] and [10]) with exactly the same experimental setup.

References

[1] Aguilar M., Barniv Y., Garrett A., *Prediction of Pitch and Yaw Head Movements via Recurrent Neural Networks*, International Joint Conference on Neural Networks, Portland, Oregon, July 2003.

[2] Egan C., Steven G., Quick P., Anguera R., Vintan L., *Two-Level Branch Prediction using Neural Networks*, Journal of Systems Architecture, vol. 49, issues 12-15, pages 557-570, December 2003.

[3] Gallant S.I., *Neural Networks and Expert Systems*, MIT Press, 1993.

[4] Laerhoven K. v., Aidoo K., Lowette S., *Real-time Analysis of Data from Many Sensors with Neural Networks*, Proceedings of the International Symposium on Wearable Computers (ISWC 2001), Zurich, Switzerland, October 2001.

[5] Mitchell T., *Machine Learning*, McGraw-Hill, 1997.

[6] Mozer M. C., *The Neural Network House: An Environment that Adapts to its Inhabitants*, Proceedings of the AAAI Spring Symposium on Intelligent Environment, Menlo Park, California, 1998.

[7] Mozer M. C., *Lessons from an adaptive house*, Smart Environments: Technology, Protocols, and Applications, J. Wiley & Sons, 2004.

[8] Petzold J., *Augsburg Indoor Location Tracking Benchmarks*, Technical Report 2004-9, Institute of Computer Science, University of Augsburg, Germany, 2004, http://www.informatik.uni-augsburg.de/skripts/techreports/.

[9] Petzold J., Bagci F., Trumler W., Ungerer T., *Context Prediction Based on Branch Prediction Methods*, Technical Report 2003-14, University of Augsburg, Germany, 2003, http://www.informatik.uni-augsburg.de/skripts/techreports/.

[10] Petzold J., Bagci F., Trumler W., Ungerer T., Vintan L., *Global State Context Prediction Techniques Applied to a Smart Office Building*, Communication Networks and Distributed Systems Modeling and Simulation Conference, San Diego, California, January, 2004.

[11] Trumler W., Bagci F., Petzold J., Ungerer T., *Smart Doorplate*, First International Conference on Appliance Design (1AD), Bristol, GB, May, 2003, Reprinted in Pers Ubiquit Comput (2003) 7: 221-226.

[12] Vintan L., Gellert A., Petzold J., Ungerer T., *Person Movement Prediction Using Neural Networks*, Technical Report 2004-10, Institute of Computer Science, University of Augsburg, Germany, 2004, http://www.informatik.uni-augsburg.de/skripts/techreports/.