Local Real-time Neural Networks-Based Learning for Tracking in an RFID Tag Field

Victor K.Y. Wu* and Nitin H. Vaidya
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Email: {vwu3, nhv}@illinois.edu

Roy H. Campbell
Department of Computer Science
University of Illinois at Urbana-Champaign
Email: rhc@illinois.edu

***Abstract*—We propose a neural networks-based learning mechanism for tracking in an RFID tag field. As users move through the field to a desired destination, they train localities of tags, creating digital trails. Later on, users seeking the destination, but without knowledge of any path, can follow the digital trails. Training information (weights from the neural networks) is stored in the tags. Our system is entirely distributed and robust to failures.**

I. Introduction

We propose a learning mechanism for tracking in a field of passive RFID tags. As mobile entities (humans or robots) equipped with RFID interrogators move through the tag field, they leave digital trails by scanning and writing to the tags. In particular, if multiple entities have a common destination and know their paths, they *train* localities of tags with respect to that destination. Later on, when a user wants to move to that destination, but does not know the path, she scans localities of tags, which will point her in the right direction.

Many technologies are available for tracking and localization, including GPS and WiFi-based methods. However, these solutions often require complicated hardware and software infrastructure, and in the case of GPS, may not even work in indoor settings. Furthermore, in emergency situations, centralized infrastructure quickly fails. In our system, we propose distributing passive RFID tags in a large area. Many researchers believe that the push for pervasive computing will result in tags being affixed to everyday objects, similar to the Internet of Things [1]. Furthermore, radios are increasingly being integrated into personal mobile devices, including RFID, in the not-too-distant future. Finally, our system is entirely distributed, robust, and improves or degrades gracefully with changes in the tag distribution. Therefore, we believe our system is ideal for a variety of tracking scenarios. Previous research have used tag fields for tracking [2], [3]. But in our system, we use tags to dynamically learn a classification.

II. System Description

Consider an indoor office building scenario, even though our ideas generalize to other situations. Passive RFID tags are distributed throughout the entire building, embedded in the floors, walls, and even ceilings. Suppose the desired destination is the cafeteria. Regular office workers know the cafeteria location, and are each equipped with an RFID interrogator. As a worker moves from her office to the cafeteria, her interrogator writes information to tags along the way, effectively marking a digital trail from her work location to the cafeteria. This is repeated with all workers, reinforcing the digital trails for a common destination. The tags form something not unlike a vector field over the building, with the cafeteria being the sink of the field lines. We call this an *RFID vector field*. Now, if a visitor equipped with an RFID interrogator wants to find the cafeteria, she can easily follow the field lines from her given location.

*This work is supported in part by NSF grant CNS-0519817.

We differentiate between *system trainers* who are the office workers updating RFID vector fields, and *system users*, who are the building visitors using those fields. System trainers and system users may interact with the tags at the same time.

III. Learning Algorithm

A. Overview

Our algorithm works on small localities of tags. For example, consider a small area with 5 tags. When a system trainer passes through this area with a desired destination, she scans the tags, and her trajectory automatically updates the 5 tags. She essentially teaches the system, indirectly, that if a user enters the area at a particular angle, then she should exit at this particular offset angle. As more system trainers pass through the area, the tags form a classification, mapping input directions to output directions. Later on, system users visiting this locality use this classification to move toward the desired destination.

B. Training

Every possible group of n tags represents a neural network, but only some of them will be of interest. Consider a system trainer passing through a group of n tags, successfully scanning them at times $t_1, \ldots, t_n$, where $t_1 \leq t_2 \leq \ldots \leq t_n$, without loss of generality. This group is "activated" at time t_n, if $t_n - t_1$ is less than some threshold time, T. The input (for training) to the group's neural network is an $n+1$-dimensional vector, $\underline{x} = (1, 0, t_2 - t_1, \ldots, t_n - t_1)$. If n is large enough for a sufficiently dense tag distribution, $\underline{x}$ is very good for characterizing the incoming trajectory angle of the system trainer. Using $\underline{x}$ as the input, the neural network produces a quantized output difference angle, $\hat{\theta}$. This is compared to the real difference angle of the system trainer's trajectory, θ, which is simply the difference in direction at time t_n, and at time t_1. (We assume θ can be easily calculated using an embedded magnetometer, frequently found in mobile devices.) The error $\theta - \hat{\theta}$ is then fed back to train the neural network. The neural network itself must be stored inline in the tags, since many system trainers are likely to train the same neural network, and they have no mechanism for communicating with each other. During the first time the neural network is trained, a unique ID that identifies the neural network and its associated RFID vector field is stored in each of the n tags. The neural network weights are stored in the tags themselves. In summary, a system trainer reads the weights at activation time, performs the training algorithm, and updates the weights in the tags. (All calculations are performed by the interrogator, with resulting weights being stored back in the tags.)

C. Testing

A system user seeking a desired destination initially moves in a straight trajectory. Whenever she scans the n tags belonging to a neural network within the threshold time, T, she reads the weights from the tags, and produces the quantized output difference angle according to the neural network. She then adjusts her trajectory by turning by that angle, and continues to move straight, until she activates the next neural network.

D. Learning Equations

We use a three-layer neural network. The input layer-to-hidden layer equations are

$$y_j = f\left(\alpha_j\right), j \in \{1, \ldots, n_{hide}\}, \tag{1}$$

where $\alpha_j = \sum_{i=0}^{n} x_i v_{ij}$, and $v_{ij}, i \in \{1, \ldots, n\}$ are the weights from the i^{th} input to the j^{th} hidden node. There are n_{hide} hidden nodes. v_{0j} are the bias weights. f is the activation function, which we take to be

$$f\left(u\right) = 1.716 \tanh\left(2u/3\right), \tag{2}$$

according to [4]. The hidden layer-to-output layer equations are

$$z_k = f\left(\beta_k\right), k \in \{1, \ldots, n_{out}\}, \tag{3}$$

where $\beta_k = \sum_{j=0}^{n_{hide}} y_j w_{jk}$, $y_0 = 1$, and $w_{jk}, j \in \{1, \ldots, n_{hide}\}$ are the weights from the j^{th} hidden node to the k^{th} output node. $k \in \{1, \ldots, n_{out}\}$. w_{0k} are the bias weights. Each k represents one of the quantized output difference angles. For example if we choose the output difference angles $\in \{0, 10, \ldots, 350\}$ degrees, then $n_{out} = 36$. The output vector is $\underline{z} = (z_1, \ldots, z_{n_{out}})$. Now suppose the target output vector is $\underline{\tilde{z}} = (\tilde{z_1}, \ldots, \tilde{z}_{n_{out}})$. During training, the component of $\underline{\tilde{z}}$ corresponding to the closest quantized output difference angle of the system trainer is set to 1. All other components are set to -1. The mean square error is $J = \frac{1}{2}\sum_{k=1}^{n_{out}} (\tilde{z}_k - z_k)$. The backpropagation algorithm adjusts the weights in a direction to reduce the error, resulting in the update equations

$$w_{jk} := w_{jk} + \eta\left(\tilde{z}_k - z_k\right) f'\left(\beta_k\right) y_j \quad \text{and} \tag{4}$$

$$v_{ij} := v_{ij} + \eta\left(\sum_{k=1}^{n_{out}} \left(\tilde{z}_k - z_k\right) f'\left(\beta_k\right) w_{jk}\right) f'\left(\alpha_j\right) x_i, \tag{5}$$

where η is the learning rate, similar to [4].

IV. System Evaluation

A. Local and Global Performances

We plan to evaluate our system both locally and globally. Locally, we want to know how well our neural network model maps input angles to output angles. We want to know how this depends on physical parameters, such as tag distribution and interrogation range. Globally, we want to know how our system behaves over an entire digital trail, ending at the desired destination. How does our system behave when there are multiple RFID vector fields? Will there be bottlenecks anywhere along the digital trails? How can we avoid them?

B. System Robustness

We plan to also investigate how robust our system is under a variety of influences. RFID tags are not always reliably scanned. They can even permanently fail. Our system is inherently robust since it is constantly being trained and updated. However, we seek to characterize this behavior. Furthermore, the system itself may significantly change, for example, if digital trails change. For example, consider a conference room being renovated. During that time, how will digital trails change, and how quickly? What are some strategies to address dynamically changing trails?

C. Simulation and Experimentation

We plan to use both simulation and experimentation in our evaluation. To rapidly evaluate local performance, we will first experimentally develop a statistical model of an interrogator's local scanning properties. Then we will use that model to simulate many neural networks with different parameters. These results will help us design the actual system. We will affix tags to floors and walls in a large indoor space. Then, we will move interrogators through this tag field, testing our system.

We will use the Motorola MC9090-G RFID handheld interrogator [5] in our experiments.

V. Conclusion

We propose a learning mechanism for tracking in an RFID tag field. We motivate our proposal, provide a system description, and detail our learning algorithm. We explain our evaluation plan. Our system is entirely distributed and very robust under a variety of conditions.

References

[1] D. Uckelmann, M. Harrison, and F. Michahelles, *Architecting the Internet of Things*, Springer, May 2011.

[2] J. Bohn, "Prototypical Implementation of Location-Aware Services based on Super-Distributed RFID Tags," in *Proc. International Conference on Architecture of Computing Systems (ARCS)*, Frankfurt, Germany, Mar. 2006, pp. 69-83.

[3] V.K.Y. Wu and N.H. Vaidya, "RFID Trees: A Distributed RFID Tag Storage Infrastructure for Forest Search and Rescue," in *Proc. IEEE Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON)*, Boston, MA, Jun. 2010, pp. 253-260.

[4] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, 2001.

[5] MC9090-G RFID Handheld Mobile Computer, *Motorola*.