6

An Artificial Neural Network Based Learning Method for Mobile Robot Localization

Matthew Conforth and Yan Meng
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ 07030, USA

1. Introduction

One of the most used artificial neural networks (ANNs) models is the well-known Multi-Layer Perceptron (MLP) [Haykin, 1998]. The training process of MLPs for pattern classification problems consists of two tasks, the first one is the selection of an appropriate architecture for the problem, and the second is the adjustment of the connection weights of the network.

Extensive research work has been conducted to attack this issue. Global search techniques, with the ability to broaden the search space in the attempt to avoid local minima, has been used for connection weights adjustment or architecture optimization of MLPs, such as evolutionary algorithms (EA) [Eiben & Smith, 2003], simulated annealing (SA) [Jurjoatrucj et al., 1983], tabu search (TS) [Glover, 1986], ant colony optimization (ACO) [Dorigo et al., 1996] and particle swarm optimization (PSO) [Kennedy & Eberhart, 1995]. The NeuroEvolution of Augmenting Topologies (NEAT) [Stanley & Miikkulainen, 2002] method turns the neural networks topology and connect weight simultaneioulsy using an evolutionary computation method. It evolves efficient ANN solutions quickly by complexifying and optimizing simultaneously; it achieves performance that is superior to comparable fixed-topology methods. In [Patan & Parisini, 2002] the stochastic methods Adaptive Random Search (ARS) and Simultaneous Perturbation Stochastic Approximation (SPSA) outperformed extended dynamic backpropagation at training a dynamic neural network to control a sugar factory actuator. Recently, artifical neural networks based methods are applied to robotic systems. In [Racz & Dubrawski, 1994], an ANN was trained to estimate a robot's position relative to a particular local object. Robot localization was achieved by using entropy nets to implement a regression tree as an ANN in [Sethi & Yu, 1990]. An ANN was trained in [Choi & Oh, 2007] to correct the pose estimates from odometry using ultrasonic sensors.

In this paper, we propose an aritifucal neural networks learning method for mobile robot localization, which combines the two popular swarm inspired methods in computational intelligence areas: Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) to train the ANN models. ACO was inspired by the behaviors of ants and has many successful applications in discrete optimization problems. The particle swarm concept originated as a simulation of a simplified social system. It was found that the particle swarm model could be used as an optimizer. These algorithms have been applied already to solving

Source: Robotics, Automation and Control, Book edited by: Pavla Pecherková, Miroslav Flídr and Jindřich Duník,
ISBN 978-953-7619-18-3, pp. 494, October 2008, I-Tech, Vienna, Austria

problems of clustering, data mining, dynamic task allocation, and optimization [Lhotska et al., 2006].

The basic idea of the proposed SWarm Intelligence-based Reinforcement Learning (SWIRL) method is that ACO is used to optimize the topology structure of the ANN models, while the PSO is used to adjust the connection weights of the ANN models based on the optimized topology structure. This is designed to split the problem such that ACO and PSO can both operate in the environment they are most suited for. ACO is ideally applied to finding paths through graphs. One can treat the ANN's neurons as vertices and its connections as directed edges, thereby transforming the topology design into a graph problem. PSO is best used to find the global maximum or minimum in a real-valued search space. Considering each connection plus one associated fitness score as orthogonal dimensions in a hyperspace, each possible weight configuration is merely a point in that hyperspace. Finding the optimal weights is thus reduced to finding the global maximum of the fitness function in that hyperspace.

The chapter is organized as follows. Section 2 introduces two swarm intelligence based methods: ant colony optimiztion and particle swarm optimization. The proposed SWIRL method is described in Section 3. Simulation results and discussions using SWIRL method for mobile robot localization are presented in Section 4. Conclusions are given in Section 5.

2. Swarm intelligence

2.1 Ant colony optimization

Dorigo et al. [11] proposed an Ant Colony Optimization (ACO). ACO is essentially a system that simulates the natural behavior of ants, including mechanisms of cooperation and adaptation. The involved agents are steered toward local and global optimization through a mechanism of feedback of simulated pheromones and pheromone intensity processing. It is based on the following ideas. First, each path followed by an ant is associated with a candidate solution for a given problem. Second, when an ant follows a path, the amount of pheromone deposit on that path is proportional to the quality of the corresponding candidate solution for the target problem. Third, when an ant has to choose between two or more paths, the path(s) with a larger amount of pheromone are more attractive to the ant. After some iterations, eventually, the ants will converge to a short path, which is expected to be the optimum or a near-optimum solution for the target problem.

2.2 Particle swarm optimization

The PSO algorithm is a population-based optimization method, where a set of potential solutions evolves to approach a convenient solution (or set of solutions) for a problem. The social metaphor that led to this algorithm can be summarized as follows: the individuals that are part of a society hold an opinion that is part of a "belief space" (the search space) shared by every possible individual. Individuals may modify this "opinion state" based on three factors: (1) The knowledge of the environment (explorative factor); (2) The individual's previous history of states (cognitive factor); (3) The previous history of states of the individual's neighborhood (social factor).

Therefore, the basic idea is to propel towards a probabilistic median, where explorative factor, cognitive factor (local robot respective views), and social factor (global swarm wide views) are considered simultaneously and try to merge these three factors into consistent behaviors for each robot. The exploration factor can be easily emulated by random movement.

3. The SWIRL approach

In the SWIRL approach, the ACO algorithm is utilized to select the topology of the neural network, while the PSO algorithm is utilized to optimize the weights of the neural network. The SWIRL approach is modeled on a school, with the ACO, PSO, and neural networks taking on the roles of administrator, teacher, and student respectively. Students learn, teachers train students, and administrators allocate resources to teachers. In the same fashion, the ACO algorithm allocates training iterations to the PSO algorithms. The PSO algorithms then run for their allotted iterations to train their neural networks. The global best score for all the neural networks trained by a particular PSO instance is then used by the ACO algorithm to reallocate the training iterations.

3.1 ACO-based topology optimization

ACO is used to allocate training iterations among a set of candidate network topologies. The desirability in ACO is defined as:

$$d(i) = \frac{1}{h+1} \tag{1}$$

where h is the number of hidden nodes in neural network i. τ (pheromone concentration) is initialized to 0.1, so that the ants' initial actions are based primarily on desirability. τ is then updated according to:

$$\tau(i,t+1) = \rho \cdot \tau(i,t) + n_a(i) \cdot \frac{g(i)}{g_{sum}} \tag{2}$$

where ρ is the rate of evaporation, n_a is the number of ants returning from neural network i, $g(i)$ is the global best for i, and g_{sum} is the sum of all the current global bests.

Each ant represents one training iteration for the PSO teacher. During each major iteration (i.e. ACO step), the ants go out into the topology space. The probability ant k goes to neural network i is given by:

$$p(i) = \frac{[\tau(i,t)]^{\alpha}[d(i)]^{\beta}}{\sum_{j=1}^{i}\left\{ [\tau(j,t)]^{\alpha} \cdot [d(j)]^{\beta} \right\}} \tag{3}$$

where α and β are constant factors that control the relative influence of pheromones and desirability, respectively.

3.2 PSO-based weight adjustment

The ANNs are then trained via the PSO algorithm for a number of iterations determined by the number of ants at that node. Each PSO teacher starts with a group of networks whose connection weights are randomly initialized. The student ANNs are tested on the chosen problem and each receives a score. The PSO teacher keeps track of the global best configuration and each student's individual best configuration. A configuration for an ANN with n connections can be considered as a point in an n+1 dimensional space, where the extra dimension is for the reinforcement score. After every round of testing, the teacher updates the connection weights of the student ANNs according to the following equations:

$$\mathbf{v}_{t+1} = c_{inr}\mathbf{r}_1 \bullet \mathbf{v}_t + c_{cgn}\mathbf{r}_2 \bullet \left(\mathbf{x}_{pb} - \mathbf{x}_t\right) + c_{scl}\mathbf{r}_3 \bullet \left(\mathbf{x}_{gb} - \mathbf{x}_t\right) \tag{4}$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1} \tag{5}$$

where position and velocity vectors are denoted by **x** and **v**, respectively. The big dot symbol is for Hadamard multiplication. The $\mathbf{r}_i$ represents vectors where each element is a new sample from the unit-interval uniform random variable. Personal best, $\mathbf{x}_{pb}$, is the point in the solution-space where that particular student received its highest score so far. Global best, $\mathbf{x}_{gb}$, is the point with the highest score achieved by any student of this PSO teacher. The three constants, c_{inr}, c_{cgn}, and c_{scl}, allow the adjustment of the relative weighting for the inertial, cognitive, and social components of the velocity, respectively.

After exhausting its allotted training iterations, the PSO teacher reports the global best to the ACO administrator. If/when it is allocated additional training iterations, the PSO teacher resumes training from exactly where it left off.

3.3 The SWIRL method summary

Pseudo-code for the SWIRL algorithm follows:

```
procedure SWIRL
        initialize ACO_Administrator
        for(candidate topology i=1…N)
                create PSO_Teacher(i)
                for(ANN_Student j=1…M)
                        initialize ANN_Student
                end for
        end for
        while(solution not found)
                compute ant movement CDF
                ants allocate training iterations
                for(PSO_Teacher i=1…N)
                        while(iterations < allocation)
                                for(ANN_Student j=1…M)
                                        test ANN_Student(j)
                                end for
                                for(ANN_Student j=1…M)
                                        update weights ANN_Student(j)
                                end for
                        end while
                        return global best
                end for
                update pheromone concentrations
        end while
end
```

4. Simulation

The SWIRL system is implemented in Java for simulation testing. There is a 5:1 ratio of ants to candidate network topologies. The candidates are 1 through 5 hidden nodes. The pheromone influence factor, desirability influence factor, and rate of pheromone evaporation are set to 2, 1, and 0.5 respectively. The initial pheromone level is 0.1 for all topologies. Each PSO teacher has 100 students. The PSO particle velocity is capped at 5. The velocity factors are 0.8 for the inertial constant, 2 for the cognitive constant, and 2 for the social constant. The neural networks are fully connected, with initial connection weights uniformly random in the range (-5,5). Hyperbolic tangent is used as the transfer function.

The SWIRL system was retrofitted into an existing Markov localization simulator, which required reimplementing SWIRL in C++. Markov localization systems are typically divided into an odometry component and a sensory component which alternate updating the belief values for each location. The SWIRL system was challenged with generating an ANN that could replace the sensory component in the Markov localization system.

Fig. 1 shows a series of snapshots of a robot localization simulation using the SWIRL system. The red pentagon is the robot. The pale blue wedges represent the robot's sonar sensors and the green rays are the robot's laser rangefinders. The large green square is the goal to which the robot must navigate. The large orange square represents the location where the robot most strongly believes itself to be. The map is divided into a grid of squares, which are each divided into a black and red triangle. The black triangle and red triangle indicate the belief of the Markov algorithm and current best ANN, respectively, that the robot may be located in that particular square. Note that the red triangles do not show up underneath the pale blue wedges; this is merely a color layering issue in the simulator's screen painter. The map is 1086 by 443 pixels in size. The Markov grid squares are 7 pixels on a side, and rotations are multiples of 10°. Note that this quantization applies only to the pose estimation in the localization system. The robot is simulated with a full range of motion using floating point x, y, and θ values. The robot has 4 sonar sensors and 19 laser rangefinders. The sonar sensor range is 75 pixels in the map, whereas the range of the laser rangefinders is 100 pixels.

Fig. 1(a) shows the localization simulator at the beginning of a simulation run. Fig. 1(b) shows an early state of the simulation. The blue pentagons are stamped periodically on the map to indicate the path taken by the robot. At this point, the robot has narrowed down its location to a few regions on the map. In Fig. 1(c), near the end of the simulation, the robot is certain of its general location. Finally, Fig. 1(d) shows the robot when it has successfully reached the goal.

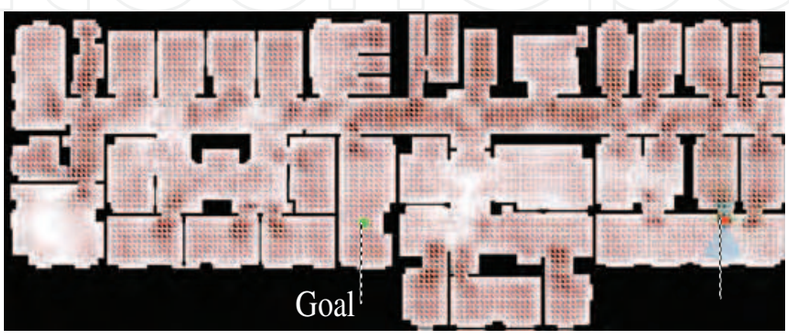

Fig. 1(a). The initialization state of a simulation.

Fig. 1(b). An early state of a simulation.

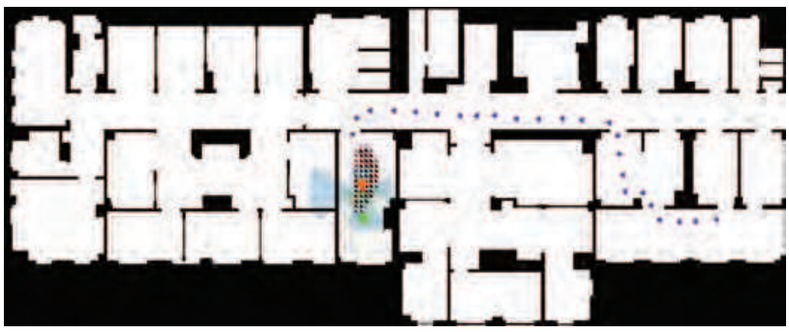

Fig. 1(c). A late state of a simulation.

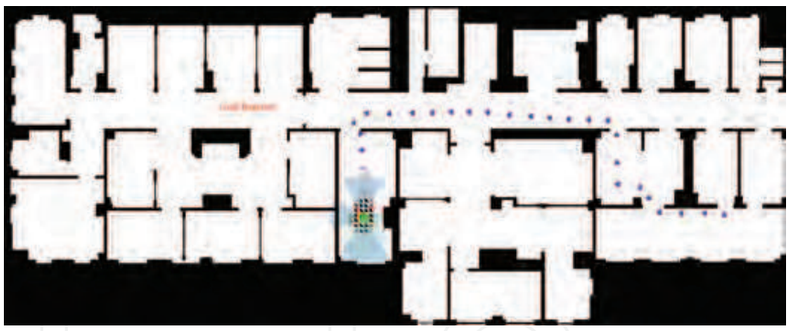

Fig. 1(d). The end of a simulation.

Fig. 1. Start of a simulation. Black = Markov value; Red = ANN value. The robot, shown as a red pentagon, must navigate from its initial position to the goal, the green square. The triangles indicate the likelihood that the robot is in that location according to the beliefs of the localization systems. The green lines show the robot's laser rangefinders and the pale blue wedges show its sonar sensors. The orange square is the robot's most likely position according to whichever localization system is currently in use for navigation. Blue pentagons are periodically stamped on the map to show the path taken by the robot.

The sensory component of the Markov localization system computes the likelihood the robot is at a particular location by comparing the current sensor readings to the predicted sensor readings for that location which are generated from the map. Sensor noise is the main source of difficulty. The following pseudo-code describes this function:

```
function probability(location x)
        prob_match=1.0
        for(i=1…Number_Sensors)
                j=normalize(reading(i)-predict(x,i))
                prob_match=prob(j)*prob_match
        end for
        return prob_match
end
```

These match probabilities are then used to update the robot's location belief matrix. In place of this, the ANN generated by SWIRL should take a vector of j's as the input, and give prob_match as the output.

The obvious source of concern in this scenario is that ANNs inherently deal in weighted sums, whereas joint probability calls for a product. This is solved by taking advantage of the fact that j > 0 and $e^{\ln(x)+\ln(y)} = xy$ for $x,y > 0$. Thus, a vector of ln(j)'s is used as the input, and e^{output} is used as prob_match.

In the simulation, a robot must navigate from an unknown starting position to a goal position that is specified on the map. Sensor noise is the main source of difficulty. The sensor noise has 3 components: bias, skew, and incidental. Each sensor has its own bias and skew values that are randomly initialized at the beginning of the simulation, but remain fixed thereafter. The incidental noise is a new Gaussian value generated each time the sensor is read.

The SWIRL system is first trained over one or more training runs. The best ANN produced is then used to replace the sensory component of the Markov localization system for the testing run. For the purpose of training, another Markov localizer with access to noise-free sensors is used to produce the "solution set" for the fitness function. This enables the ANN (ideally) to weigh the sensors according to their relative accuracy. The simulation results are shown in Fig. 2, Fig. 3, and Fig. 4. The results are produced for 1 and 3 training runs.

Unfortunately, the wavefront navigation system used by the Markov localization simulator introduces substantial variability into the robot's performance, as demonstrated by the large standard deviations. Consequently, it would require an exorbitantly large number of tests to establish with a high degree of certainty whether the average SWIRL solution is slightly better or slightly worse than the Markov method. In any case, improving Markov localization is not the goal here. It is clear from the results that SWIRL can indeed produce ANNs that function comparably to the sensory component of the Markov localization system.

SWIRL demonstrates its ability to generate effective solutions in the face of real-world complications such as noise and poor calibration. After only a single test run, the SWIRL solution is already performing comparably to the Markov method. The reason more fine-grained progress is not presented is that the nature of Markov localization is such that early mistakes are compounded each cycle. No matter how good the SWIRL solution gets, its mistakes from earlier in the run will prevent it from making accurate predictions. To make the graphical display useful, every so often during training (**only** during training) SWIRL's old belief matrix is overwritten with "correct" values so that one can see an estimate of how accurate the best ANN is at that point. However, this obviously gives a rough estimate only, and is not appropriate for a numerical comparison with the Markov predictions.

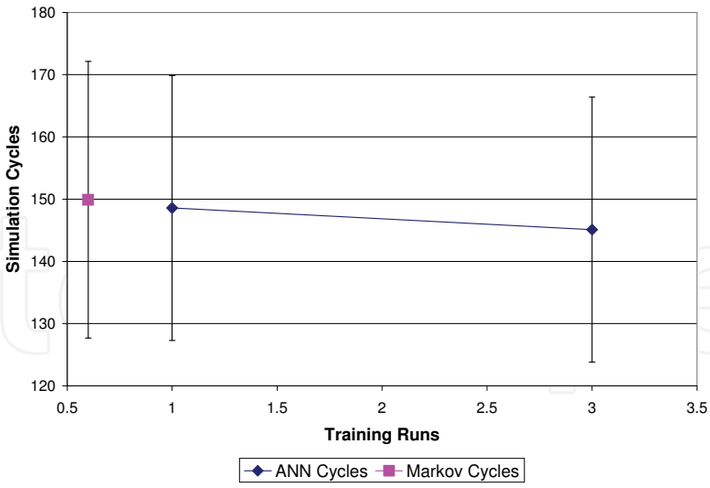

Fig. 2. Simulation cycles versus training runs.

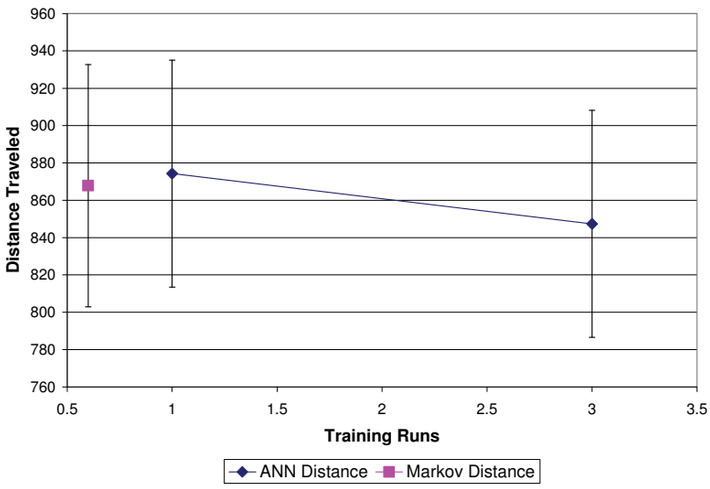

Fig. 3. Distance traveled versus training runs

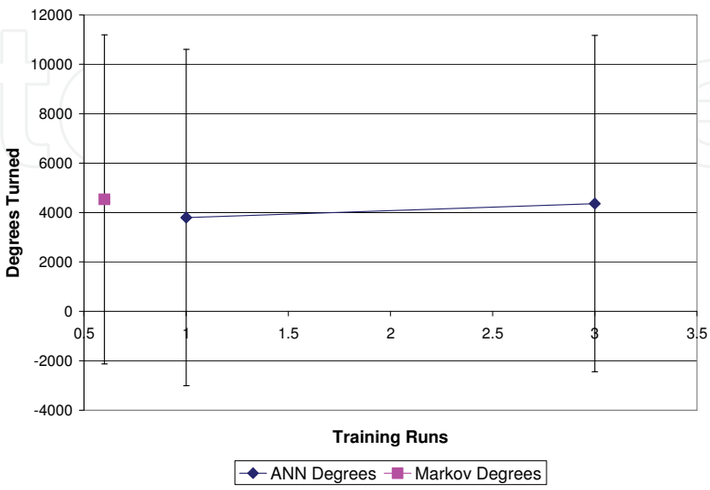

Fig. 4. Degrees turned versus training runs

7. Conclusion

In this paper, a SWIRL algorithm is proposed to generate ANN solutions to tasks/problems amenable to reinforcement learning. Basically, the ACO algorithm is applied to select the neural network topology, while the PSO algorithm is utilized to adjust the connection weights of the selected topology. The robot localization using the SWIRL algorithm have been efficiently conducted in a noisy, imprecise environment with large-scale number of input signals, as might be encountered in the real world. As a result of its generality, the SWIRL method is scalable, robust, and can be applied to almost any real world task. This implementation is merely a simple proof-of-concept where the ACO algorithm chooses the number of hidden nodes. By limiting the scope to fully-connected feedforward neural networks, this single number fully defines the topology (since the input and output nodes are determined by the test problem). These are limits of this test implementation only; not limits on the SWIRL method.

8. References

Haykin, S. (1998). *Neural Networks: A comprehensive Foundation*. 2nd Edition, Prentice Hall.

Eiben, E. & and Smith, J.E. (2003). *Introduction to Evolutionary Computing.* Natural Computing Series. MIT Press. Springer. Berlin.

Kirkpatrick, S.; Gellat Jr., C.D., & Vecchi, M. P., (1983). Optimization by simulated annealing, *Science*, 220: 671-680.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence, *Computers and Operation Research*, Vol. 13, pp. 533-549.

Dorigo, M.; Maniezzo, V., & Colorni. A. (1996). Ant System: optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man and Cybernetics - Part B*, vol. 26, no. 1, pp. 29-41.

Kennedy, J. & Eberhart, R. (1995). Particle Swarm Optimization, in: *Proc. IEEE Intl. Conf. on Neural Networks (Perth, Australia)*, IEEE Service Center, Piscataway, NJ, IV:1942-1948.

Stanley, K. O. & Miikkulainen. R. (2002). Evolving Neural Networks through Augmenting Topologies, *Evolutionary Computation,* 10(2): 99-127.

Patan, K. & Parisini, T. (2002). Stochastic learning methods for dynamic neural networks: simulated and real-data comparisons, *Proceedings of American Control Conference*.

Racz. J. & Dubrawski, A. (1994). Mobile Robot Localization with an Artificial Neural Network, International Workshop on Intelligent Robotic Systems IRS '94, Grenoble, France.

Sethi, I. K. & Yu, G. (1990). A Neural Network Approach to Robot Localization Using Ultrasonic Sensors, *Proceedings of 5th IEEE International Symposium on Intelligent Control*, 1990. pp. 513-517 vol. 1, 5-7.

Choi, W. S. & Oh, S. Y. (2007). Range Sensor-based Robot Localization Using Neural Network, *International Conference on Control, Automation and Systems*, pp. 230-234, 17-20.

Lhotská, L.; Macaš, M., & Burša, M. (2006). *PSO and ACO in Optimization Problems*, E. Corchado et al. (Eds.): IDEAL 2006, LNCS 4224, pp. 1390 – 1398.

Robotics Automation and Control

Edited by Pavla Pecherkova, Miroslav Flidr and Jindrich Dunik

ISBN 978-953-7619-18-3
Hard cover, 494 pages
Publisher InTech
Published online 01, October, 2008
Published in print edition October, 2008

This book was conceived as a gathering place of new ideas from academia, industry, research and practice in the fields of robotics, automation and control. The aim of the book was to point out interactions among various fields of interests in spite of diversity and narrow specializations which prevail in the current research. The common denominator of all included chapters appears to be a synergy of various specializations. This synergy yields deeper understanding of the treated problems. Each new approach applied to a particular problem can enrich and inspire improvements of already established approaches to the problem.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Matthew Conforth and Yan Meng (2008). An Artificial Neural Network Based Learning Method for Mobile Robot Localization, Robotics Automation and Control, Pavla Pecherkova, Miroslav Flidr and Jindrich Dunik (Ed.), ISBN: 978-953-7619-18-3, InTech, Available from: http://www.intechopen.com/books/robotics_automation_and_control/an_artificial_neural_network_based_learning_method_for_mobile_robot_localization

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen