Global Self-Localization for Autonomous Mobile Robots Using Region- and Feature-Based Neural Networks

Jason A. Janét, Ricardo Gutierrez-Osuna, Troy A. Chase, Mark White and Ren C. Luo
Center for Robotics and Intelligent Machines
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695-7911

Abstract - *This paper presents an approach to global self-localization for autonomous mobile robots using a region- and feature-based neural network. This approach categorizes discrete regions of space using mapped sonar data corrupted by noise of varied sources and ranges. Our approach is like optical character recognition (OCR) in that the mapped sonar data assumes the form of a character unique to that room. Hence, it is believed that an autonomous vehicle can determine which room it is in from sensory data gathered while exploring that room. With the help of receptive fields, some pre-processing, and a robust exploration routine, the solution becomes time-, translation- and rotation-invariant. The classification rate of this approach is comparable to the Kohonen based approach. Some pros and cons of both approaches will be discussed.*

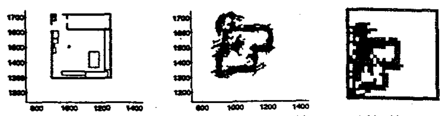
1. Introduction

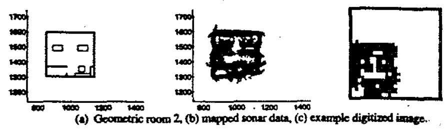
Can a robot determine which region of space it is in without knowing how it got there? To date a significant amount of work has been devoted to developing *low-level* self-localization approaches for autonomous mobile robots. These approaches depend on prior dead reckoning estimates and discrete-time models that iteratively rationalize and correct robot position and orientation based on correlations between predicted and actual sensor data [4, 5, 7, 11, 19-23, 28-30]. But, without an *initial* reliable position estimate, even the most proven techniques can become ineffective.

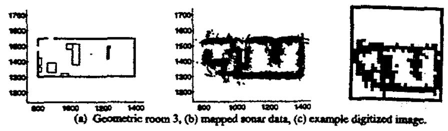
The objective of this research is to endow autonomous mobile robots with the ability to perform self-localization on a *global* level. That is, the robot should be able to use sensor data to determine which region of indoor space it is in. See figure 1. Since most indoor environments can be easily segmented into *rooms*, different room configurations will define discrete regions of space. Hence, the global self-localizer is expected to learn features unique to a room and associate sensor data sets with the previously learned rooms.

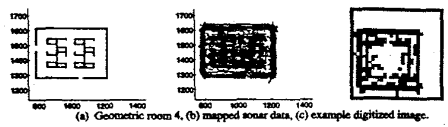
1.1 Time-, Translation- and Rotation-Invariance

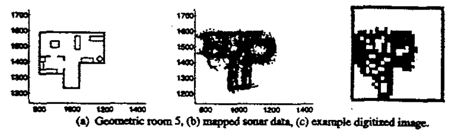
A robust global self-localization (GSL) technique should have the following characteristics. First, it should be time-invariant simply because no two robots will explore a room the same way. In fact, a single robot will likely not follow the same trajectory each time. Second, it should be translation-invariant because the robot does not know the actual global coordinates of the region of space it is in, much less the sensor data it collects. Third, it should be rotation-invariant because, through the course of becoming lost, a robot can also become disoriented.

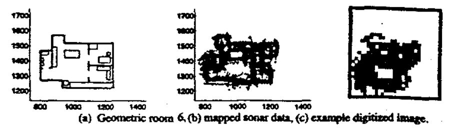

(a) Geometric room 1, (b) mapped sonar data, (c) example digitized image.

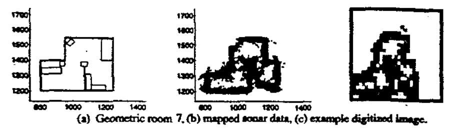

(a) Geometric room 2, (b) mapped sonar data, (c) example digitized image.

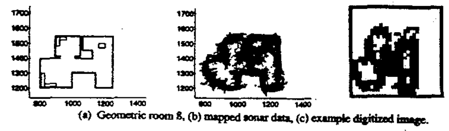

(a) Geometric room 3, (b) mapped sonar data, (c) example digitized image.

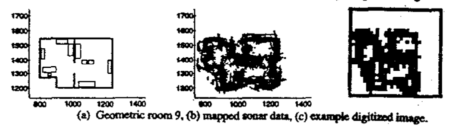

(a) Geometric room 4, (b) mapped sonar data, (c) example digitized image.


(a) Geometric room 5, (b) mapped sonar data, (c) example digitized image.


(a) Geometric room 6, (b) mapped sonar data, (c) example digitized image.


(a) Geometric room 7, (b) mapped sonar data, (c) example digitized image.


(a) Geometric room 8, (b) mapped sonar data, (c) example digitized image.


(a) Geometric room 9, (b) mapped sonar data, (c) example digitized image.

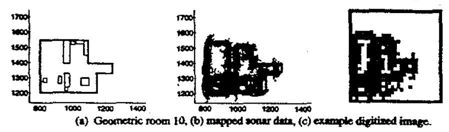

(a) Geometric room 10, (b) mapped sonar data, (c) example digitized image.

Figure 1. Geometric rooms, sonar data and digitized image.

1.2 Region- and Feature-Based Neural Networks

Several applications of solving the optical character recognition (OCR) problem with neural networks have been investigated [10, 12, 13, 26, 27, 31, 34, 35]. Kohonen neural networks were used to treat the GSL problem as an OCR problem with great success [23]. In this application, a region- and feature-based neural network (RFNN) recognizes the *room character* associated with a particular room. The RFNN is a multi-layered neural network which uses receptive fields and weight sharing [16]. In general, multi-layered neural networks are known for their abilities to perform classification, recognition, data compression and association [14, 16, 18, 31, 35]. The only information that must be estimated beforehand is which regions of the problem space (room character image) are important and the maximum expected number of features per region.

2 The Domain

We assume the environment to be in R^2 space. Specifically, as a robot wanders autonomously or is manually driven around a room, its sensor data is mapped to a two-dimensional plane. Since ultrasonic sensors seem to be the sensor of choice for autonomous and semi-autonomous vehicles, it is felt that the sensor used for extracting feature information should also be the ultrasonic sensor [1, 2, 6, 9, 24, 25, 32].

2.1 The Autonomous Mobile Robot Simulation

Given that the simulation described in [19-23] has proven accurate at modeling both sonar and robot behavior and that it provides a graphical display essential to understanding each step of solving this problem, it was considered a suitable platform for creating training sets and test sets. The simulated sonar models are based on work done by Kuc [24], Moravec [32] and Barshan [2] as well as information provided by Polaroid [33] and Cybermotion [8].

Also, the simulation can ensure that the 160 training sets and 180 test sets could be created in a timely fashion and without the risk of harming either the robot or the environment. Furthermore, the simulation can guarantee that no furniture is radically rearranged through the course of creating the training and test sets. Finally, the simulation can track both the robot's *dead reckoning* coordinates (i.e., where the robot *thinks* it is) and the robot's *actual* coordinates (i.e., where the robot *really* is) without needing someone to physically measure and/or estimate the true location of the real robot.

2.2 Room Character Generation

A room's unique character is created by clouds of sonar data points collected and mapped by the robot in its travels through the region. Typically, these clouds are clustered near the geometric beacon surfaces encountered by the sonar windows. With a single-transducer sonar it is difficult to know from a TOF reading the specific point that produced an echo because sonars sample a *region*. Hence, we assume that each sonar reading occurs along the axis normal to its transducer ($X_S = 0$, $Y_S =$ TOF). To map the TOF reading, the point is transformed from the sonar frame to the global frame. TOF readings are mapped according to where the robot *thinks* it is (i.e., dead reckoning).

2.2.1 Sonar Corruption. Sonar readings are inherently corrupted by noise. So, "to represent the random errors in an adequate manner for a properly controlled experiment", a 6σ Gaussian corruption function is applied to all TOF readings calculated by the simulation [17]. Specifically, for a *calculated* TOF distance, R_{calc}, C_s is a user-specified variable in the closed interval $C_s \in [0,1]$ that defines the range of noisy readings by R_{sens}. See figure 2.

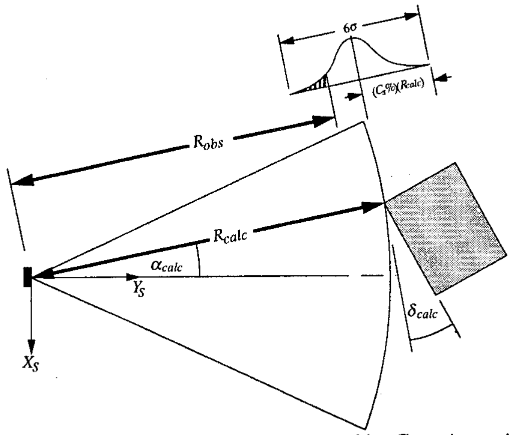


Figure 2. Simulated sonar reading corrupted by Gaussian noise.

2.2.2 Trajectory Corruption. The very motivation for developing low-level sensor-based self-localization techniques stems from the fact that true mobile robot dead reckoning (from odometers, inertial navigation systems, etc.) grows more and more unreliable with each bump in the floor and turn of the robot. That is, where the robot *thinks* it is might not be where it *actually* is. To simulate this phenomenon, a user-defined random Gaussian trajectory corruption is induced as shown in figure 3.

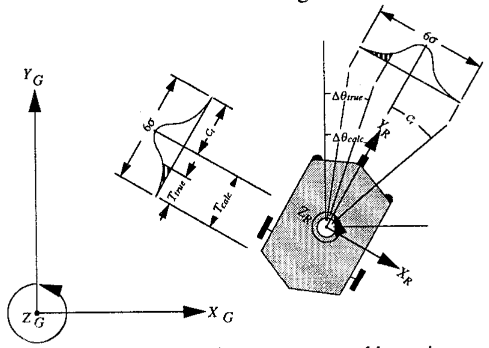


Figure 3. Robot trajectory corrupted by noise.

A 6σ Gaussian corruption function is applied to all robot rotations and translations. Rotation corruption is limited to the closed interval $C_r^\circ \in [-\pi, \pi]$ and is specified by the user to define the range of potential noise added to each turn. Translation corruption is a variable on the closed interval $C_t \in [-1,1]$ and is also specified by the user to define the range of potential noise added to each translation. Figure 4 shows mapped sonar data from a simulated robot subjected to different degrees and types of corruption.

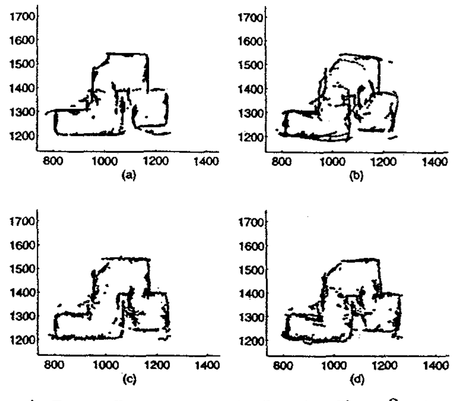


Figure 4. Room 7 character: a) $C_s = 0,\ C_r^\circ = 0^\text{o},\ C_t = 0$; b) $C_s = 0,\ C_r^\circ = 18^\text{o},\ C_t = 10\%$; c) $C_s = 15\%$, $C_r^\circ = 0^\text{o},\ C_t = 0$; d) $C_s = 15\%,\ C_r^\circ = 18^\text{o},\ C_t = 10\%$.

3 Region- and Feature-based Neural Networks

The RFNN software is a high-level program that provides a flexible, multi-layered, feed-forward architecture as well as the capability to add to and prune from the architecture even after training has begun. The user simply specifies the desired architecture, initialization parameters, learning parameters and stop-training conditions. Figure 5 shows a typical multi-layered, feed-forward neural network that is ideal for back-prop learning.

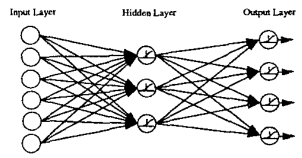
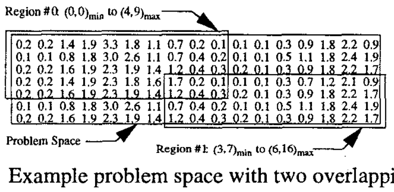


Figure 5. Multi-layered, feed-forward architecture.

3.1 Regions: Input-to-Hidden Neuron Connectivity

The input layer is where the problem space is seen by the neural network. The problem space can be broken down into two-dimensional *regions*. Regions can overlap each other or define independent portions of the problem space. Figure 6 shows an example problem space with two overlapping regions.


Region #0: $(0,0)_{min}$ to $(4,9)_{max}$
Problem Space
Region #1: $(3,7)_{min}$ to $(6,16)_{max}$

0.2	0.2	1.4	1.9	3.3	1.8	1.1	0.7	0.2	0.1	0.1	0.1	0.3	0.9	1.8	2.2	0.9
0.1	0.1	0.8	1.8	3.0	2.6	1.1	0.7	0.4	0.2	0.1	0.1	0.5	1.1	1.8	2.4	1.9
0.2	0.2	1.6	1.9	2.3	1.9	1.4	1.2	0.4	0.3	0.2	0.1	0.3	0.9	1.8	2.2	1.7
0.2	0.2	1.4	1.9	2.3	1.8	1.6	1.7	0.2	0.1	0.1	0.1	0.3	0.7	1.2	2.1	0.9
0.2	0.2	1.6	1.9	2.3	1.9	1.4	1.2	0.4	0.3	0.2	0.1	0.3	0.9	1.8	2.2	1.7
0.1	0.1	0.8	1.8	3.0	2.6	1.1	0.7	0.4	0.2	0.1	0.1	0.5	1.1	1.8	2.4	1.9
0.2	0.2	1.6	1.9	2.3	1.9	1.4	1.2	0.4	0.3	0.2	0.1	0.3	0.9	1.8	2.2	1.7

Figure 6. Example problem space with two overlapping regions.

With regards to the neural network architecture, the arrangement of regions defines the connectivity between the input layer and hidden layer neurons. Specifically, each region has its own dedicated hidden layer. Hence, if the desired architecture were to require fully connected input-to-hidden neurons, there would be only one region in the entire problem space. Otherwise, the input-to-hidden neurons could be made partially connected by breaking the problem space into more than one region. See figure 7.

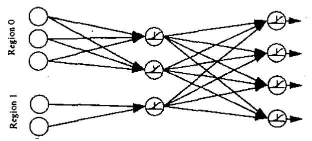


Figure 7. Regions define input-to-hidden neuron connectivity.

3.2 Features: Hidden Neuron Convolution Kernels

We use the term *feature* because the RFNN bases its final output decision on the quantity and/or location of certain characteristic features unique to an overall larger class of objects, images or patterns. Such features might be shared by several classes. But each unique class might be characterized by a certain number of particular feature occurrences, or that particular features be found in specific locations of the region. For example, a dial clock and a rotary phone might both have numbers on them, but a clock will have *more* numbers than the phone. Further, the *locations* of particular numbers on the phone are different from the *locations* of the same numbers on the clock.

A *feature* is defined as a movable hidden layer neuron with a unique set of input-to-hidden layer synapses. The number of hidden layer neurons for a given region is dictated by the number of features it has. The set of input-to-hidden layer synapses (receptive field) unique to a feature is called the *feature weight matrix* (FWM).

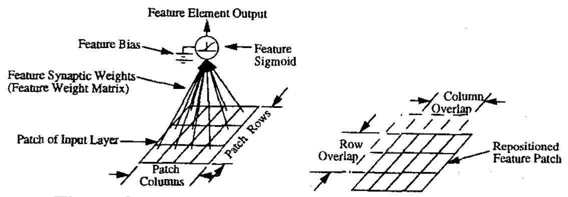


Figure 8. Feature neuron, FWM, patch and overlap.

The FWM is convolved with a region to search for a possible feature match. This convolution process is similar to morphological operations in computer vision where a kernel is compared to all parts of an image to look for a pattern match [15]. The number of input-to-hidden layer synapses in each FWM is defined by the feature *patch* size (kernel size). Specifically, a patch is a subset of its respective region and, hence, its size is defined by a number of rows and columns less than or equal to the region's minimum and maximum extremes inside the problem space.

The convolution operation is like sliding a window over an image where, for each position of the window, the portion of image inside that window can be measured by the feature neuron for similarity to a previously learned feature. Placement of each feature patch depends on the *overlap* parameters. Specifically, the overlap size defines the amount of rows and columns a patch can be overlapped when it is repositioned. Patch overlapping can compensate for slight rotations and translations of a feature by searching areas of the region *at* and *near* the location of the expected feature occurrence.

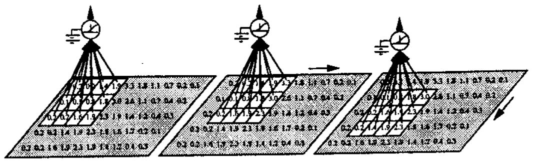


Figure 9. Feature patch and neuron being convolved with image.

In general, we want to minimize the number of features to minimize the overall architecture data size. Choosing the optimal number of features each region can prove to be a rather subjective and arbitrary process. One might decide that, for example, the RFNN will have to look for horizontal and vertical lines and thus decide that only two features are necessary. In truth, however, this way of envisioning how the RFNN learns a feature might be completely different from how it actually learns a feature. After all, unlike computer vision morphology, the neural network decides for itself how a feature should be measured. This is because the RFNN not only looks at *whether* a candidate feature does or does not completely match; it also looks at the *degree* of match.

3.3 Hidden-to-Output Connectivity with the RFNN

For each position of a feature inside its region, a unique value associated with that position is produced by the feature neuron. These position-dependent feature values are placed in a *feature element matrix* (FEM) where they are held and later presented to the output neuron layer. Every feature has its own FEM. The general idea behind the FEM is to let the output layer know where and to what degree certain features were found. Essentially, this creates the illusion that there are many more hidden neurons in the architecture since there is a hidden neuron for each position (FEM element) of a feature patch. In truth, however, this is not the case since a single feature will use the same input-to-hidden synaptic weights at each position of its patch. Hence, synapses exist between the FEM elements and the output neurons. Figure 10 shows how a FEM is filled and then connected to an output layer neuron.

The set of FEM element-to-output weights is different for each output neuron. This is because each output neuron is entitled to place as much or as little significance on a given type of feature. After all, not all feature types can be expected to be relevant to all output neurons. If the FEM element-to-output synaptic weights are matured independent of each other, it is said that these weights are *uncoupled.* Uncoupled weights presumably let an output neuron know where and to what degree a particular feature matches the input data. If, on the other hand, the FEM element-to-output synaptic weights are matured so that they are all equal to each other for a particular output neuron, it is said that these weights are *coupled.* Coupled weights presumably let an output neuron know *how many times* a feature matches.

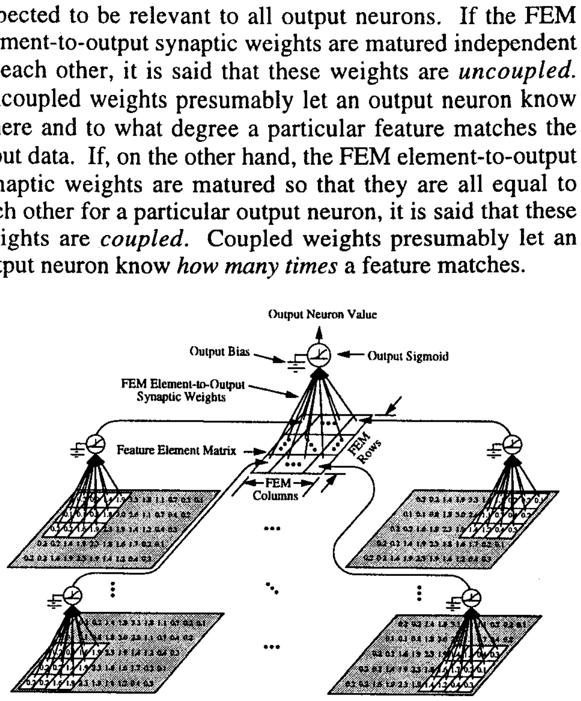


Figure 10. Connectivity of a FEM's elements to an output neuron.

4 Strategy

Single-region, multiple-feature, ten-output RFNN's will be used to learn the relevant features common and unique to each of the ten rooms. The input problem space is an image of 29x29 binary pixels that cover an area 16 meters wide and 12.7 meters long. The presence of a sonar point inside a pixel's boundaries activates that pixel. Four different architectures will be examined: the first will use only one feature (sub-image); the second will use two features; the third will use three features; and the fourth will use four features. Common to all architectures will be feature-patch size (sub-image size of 5 rows by 5 columns), maximum overlap (4 rows and 4 columns) and that the entire problem space is viewed by the RFNN through a single region. To test the capabilities of the mature RFNN, test data sets will be collected by the simulated robot as it autonomously drives around the room. Generality will be tested by allowing test data corruption ranges to be larger than those used for training. Two pre-processing operations will be applied to digitize the room character and reduce the impact of large-scale translation variance.

4.1 Training and Testing Data

In a realistic setting, a robot can collect data while either being manually driven around the room (manual teaching) or autonomously driving through certain sections of the room while carrying out other tasks. Trajectories and explored features vary from data set to data set. However, one extremely important requirement of collecting training and testing data is that the robot explore (what it perceives to be) the southwest extremes of each room. This aspect helps make the GSL problem translation-independent. That is, to transpose the mapped sonar data as close to a common reference frame origin as possible, we need mapped data that reflects minimum x- and minimum y-coordinate values (*west* and *south* respectively) a room might have.

4.2 Training and Recall

The RFNN is trained with error back-propagation. To expedite the learning process, all synaptic weights in the architecture undergo a greedy version of *adaptive learning* [16]. That is, the learning rates, unique to each synapse, vary according to the history of error reduction. The "greedy" part implies that each synapse's learning rate can become greater than one. Learning rates and weights are updated at each *epoch*, and an epoch is considered to be one full training pass through all of the training patterns.

Testing the capabilities of this RFNN-based global self-localization approach is a matter of presenting newly acquired room data to the mature neural network. Since each output neuron is unique to a room, recall is simply a check to see if the output neuron with the highest output value corresponds to the index of the correct room.

4.3 Pre-Processing

Two pre-processing procedures are implemented in this research. First, *gross shifting* is used in all phases of training and testing to approximate a translationally-invariant problem space [23]. This procedure crudely fits a sonar data set arbitrarily placed in 2D space to a reference frame origin common to all rooms and datasets, also arbitrarily placed in 2D space. As figure 11 shows, minimum x- and y-values of the data set are subtracted from all data points, to transpose it to a common reference frame origin. While this takes care of the major portion of the translation-invariance problem, it can be seen that a single outlier in either the x- or y-direction can still produce slight translation variances between the sonar data and a room. It is for this reason that we use receptive-fields architectures.

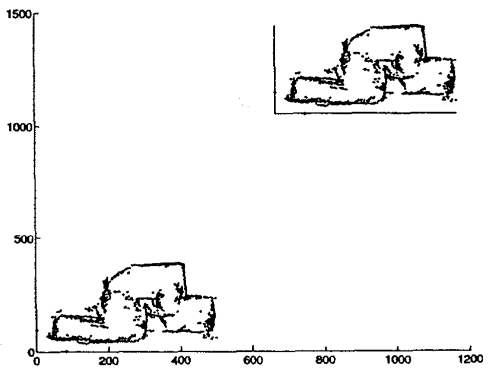


Figure 11. Gross shifting a room character data set (placed arbitrarily in 2D space) to a common reference frame.

The second pre-processing step, *digitization*, is used to discretize the problem space into a binary image of pixels. A pixel activated by the presence of a mapped sonar echo point will have the value of one. All pixels that are not activated will maintain a value of zero.

5 Experimental Results/Concluding Remarks

Shown in figure 12, all four candidate architectures yielded classification rates better than 96.67%. Of the four, the single-region, 3-feature, 10-output architecture yielded the highest classification rate, 98.89%. Of course, there is a tradeoff between data size and classification rates. After all, the discrepancy between the lowest and highest classification rates might actually be trivial when compared to the amount of RAM needed for the larger architecture. What is truly interesting, however, is that only a single hidden layer neuron with a 5x5 patch was necessary to properly classify more than 96% of the 180 test sets from 10 different rooms. Even the best architecture seen in this research needed only three hidden layer neurons.

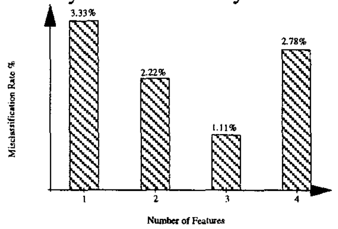


Figure 12. Misclassification rates of single region, variable-feature, ten-output architectures.

5.1 Comparison with Kohonen-based GSL

Significantly less pre-processing time is required for the RFNN when compared to what was required to get a 98.89% classification rate with the Kohonen neural network reported in [23]. That is, the RFNN required only two simple pre-processing steps (gross-shifting and digitization) whereas the Kohonen neural network required three (gross-shifting, fine shifting and dormant node pruning). The fine shifting, in particular, was extremely time consuming because, for each data set, it forces the Kohonen to do recall several times on sampled and incrementally shifted sonar data until a *best fit* between the Kohonen mesh and sonar data is found. Without such fine shifting, the Kohonen neural network could not compensate as well for small-scale translation descrepancies (unlike the receptive fields).

The Kohonen is not without its merits, however. First, the Kohonen did not require *a priori* knowledge of the problem space size. The RFNN presented in this paper not only requires that we make an assumption on the maximum dimensions of the problem space, but also that we assume an adequate pixel size (resolution). Like most computer vision principles (threshold, kernel size, etc.), predicting a necessary field of view and/or resolution for a digitized room character can be rather subjective. Second, in many ways the Kohonen is much easier (albeit more time consuming) to train. For example, the Kohonen nodes are easily initialized to increments scaled to the data set size. The RFNN, however, requires that the synaptic weights be initialized to values small enough to prevent saturation. The Kohonen also proved tolerant of a wide range of learning rates and neighborhood sizes. The RFNN seemed rather sensitive to initialized synaptic weight learning rates outside a small spectrum (between 0.002 and 0.004). Certainly, optimization schemes exist that find the best initialized weights and learning rates to prevent saturation [16].

5.2 Comparison with OCR and Computer Vision

The two multi-layered neural networks that have been suggested for optical character recognition by LeCun and Säckinger have more than one hidden layer [16, 34]. In fact, the OCR network suggested by Säckinger has four hidden layers, a 20x20 binary input problem space and 10 outputs which are used to decide which of the first ten positive integers (0, 1, ..., 9) are handwritten inside the problem space. The overall architecture of this OCR network has more than 130,000 hidden layer synaptic connections and more than 3,000 output synaptic connections which undoubtedly consume considerable RAM. The OCR network also uses receptive fields, weight sharing and patch overlap to degrees that are unique to each hidden layer.

Although the room characters that the RFNN learned were not integers, they were characters none-the-less. So this common ground exists between the OCR and RFNN networks. The problem space for the OCR network (20x20) is smaller than that for the RFNN (29x29). But we can still conclude that, if the RFNN has fewer synaptic weights than the OCR, the data size would be even smaller if the problem space for the RFNN was 20x20. The largest RFNN architecture examined in this paper (four features, 5x5 patch), was comprised of 100 input-to-hidden synapses and 25,000 FEM element-to-output synapses. The OCR network requires 130,000 hidden layer synapses and 3,000 hidden-to-output synapses. The question is, of course, can the RFNN perform as well as the OCR network on the same handwritten data? This is left to future research.

In general, the RFNN seems to be a cross between computer vision morphology [15] and the OCR network. The most apparent difference between the RFNN and computer vision morphology is that the RFNN decides *autonomously* what the convolution kernel looks for.

5.3 General Comments

Of the four architectures, the best classifier (98.89%) utilized gross shifting, digitization and a single-region, three feature architecture. However, the difference between classification rates for the four different architectures is not all that large. This is for a static environment; the effects of other unknown objects on the recognition have not yet been tested. A radical redesign of a room would require any system to relearn the room. The addition of new furniture or the rearrangement of existing furniture will change the signature character of the room. How much the character changes and the effect it has on recognition is situational. This suggests the need for an update learning scheme that notices small physical changes in a previously trained room.

References

[1] B. Barshan and R. Kuc, "Differentiating Sonar Reflections from Corners and Planes by Employing an Intelligent Sensor." *IEEE Trans. on Pattern Analysis and Machine Intelligence* Vol. 12 no. 6, June 1990, pp. 560-569.

[2] B. Barshan and R. Kuc, "Active Sonar for Obstacle Localization Using Envelope Shape Information." *Proc. 1991 Int'l Conf. on Acoustics, Speech and Signal Processing*, 1991, pp. 1273-1276.

[3] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots." *IEEE Trans. on Rob. and Aut.* Vol. 7 no. 3, June 1991.

[4] C. Brown, H. Durrant-Whyte, et al., "Centralized and Decentralized Kalman Filter Techniques for Tracking, Navigation and Control." in *Proc. DARPA Image Understanding Workshop*, 1989.

[5] C. Brown, H. Durrant-Whyte, et al. (1989). Kalman filter algorithms, applications and utilities. Oxford University Robotics Research Group.

[6] O. Bozma and R. Kuc, "Building a Sonar Map in a Specular Environment Using a Single Mobile Sensor." *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 13 no. 12, pp. 1260-1269, Dec 1991.

[7] I. J. Cox and J. J. Leonard, "Probabilistic Data Association for Dynamic World Modeling: A Multiple Hypothesis Approach." in *Fifth International Conference on Advanced Robotics*, 1991, pp. 1287-1294.

[8] Cybermotion, User's Manual. 5457 Aerospace Road; Roanoke, VA.

[9] A. Elfes, "Sonar-Based Real-World Mapping and Navigation." *Journal of Robotics and Automation* Vol. 3 no. 3, June 1987, pp. 249-265.

[10] E. G. Elliman, R. N. Banks, "Shift Invariant Neural Net for Machine Vision." *IEE Proceedings*, Vol. 137, Pt. I, No. 3, June 1990, pp. 183-187.

[11] H. R. Everett, G. A. Gilbreath, et al. (1990). Modeling the Environment of a Mobile Security Robot. *Tech. Doc. 1835*. San Diego, CA, Naval Ocean Sys Ctr.

[12] K. Fukushima, S. Miyake, T. Ito, "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition." *IEEE Trans.on Systems, Man, and Cybernetics*, Vol SMC-13, No. 5, Sept 1983, pp 826-834.

[13] K. Fukushima, "A Neural Network for Visual Pattern Recognition." *IEEE Computer*, March 1988, pp 65-75.

[14] S. I. Gallant, *Neural Network Learning and Expert Systems*, MIT Press, Cambridge, MA, 1993, pp. 136-143.

[15] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision: Volumel*, Addison-Wesley Publishing Co., Reading, MA, 1992.

[16] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Co., NY, 1994.

[17] J. P. Holman, *Experimental Methods for Engineers*, 5th ed., McGraw-Hill Book Company, NY, 1989, pp. 57-61.

[18] R. Holdaway , M. White, "Computational Neural Networks: Enhancing Supervised Learning Algorithms via Self-Organization." *Int. J. Biomed Comput.*, 1990, pp. 151-167.

[19] J. A. Janét, R. C. Luo, et al., "Sonar Windows and Geometrically Represented Objects for Mobile Robot Self-Referencing." *IEEE/RSJ International Conference on Intelligent Robotics and Systems*, 1993.

[20] J. A. Janét, S. G. Goodridge, Ren C. Luo, "Dynamic Motion Control of Sensor-Based Autonomous Mobile Robot." *APS International Conference on Mechatronics and Robotics*, April 1994, Aachen, Germany.

[21] J. A. Janét, R. C. Luo, M. G. Kay, "Autonomous Mobile Robot Motion Planning Enhanced with Extended Configuration-Space and Half-Planes." *IEEE/SICE/AEI Int. Conf. on Industrial Electronics*, Sept. 1994, Bologna, Italy.

[22] J. A. Janét, R. C. Luo, M. G. Kay, "Traversability Vectors Make Autonomous Mobile Robot Motion Planning and Self-Referencing More Efficient." *IEEE/RSJ Int'l Conf on Intelligent Robotics and Systems*, Sept. 1994.

[23] J. A. Janét, R. Gutierrez-Osuna, et. al., "Global Self-Localization for Autonomous Mobile Robots Using Self-Organizing Kohonen Neural Networks." *IEEE/RSJ Int'l Conf on IROS*, Aug. 1995, Pittsburgh, PA.

[24] R. Kuc, "A Spatial Sampling Criterion for Sonar Obstacle Detection." *IEEE Trans. on Pattern Analysis and Machine Intelligence* Vol. 12 no. 7, July 1990.

[25] R. Kuc and V. Viard, "A Physically Based Navigation Strategy for Sonar-Guided Vehicles." *Int. Journal of Robotics Research* Vol. 10 no. 2, April 1991.

[26] Y. LeCun, B. Boser, et. al. "Backpropagation Applied to Handwritten Zip Code Recognition." *Neural Computation*, Vol 1, 1989, pp 541-551.

[27] Y. LeCun, B. Boser, et. al. "Handwritten digit recognition with a back-propagation network." *Advances in Neural Information Processing Systems 2*, (D. S. Touretsky, ed.) pp 396-404, San Mateo, CA: Morgan Kaufmann, 1990.

[28] J. Leonard, H. Durrant-Whyte, et al., "Dynamic Map Building for an Autonomous Mobile Robot." in *IEEE Int. Conf. on IROS*, 1990.

[29] J. Leonard and H. Durrant-Whyte, "Mobile Robot Localization by Tracking Geometric Beacons." *Trans on Rob. and Aut.* Vol. 7 no. 3, June 1991.

[30] J. Leonard and H. Durrant-Whyte, *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers. Norwell, Massachusetts. 1992.

[31] R. P. Lippman, "An Introduction to Computing with Neural Nets." *IEEE ASSP Magazine*, April 1987, pp. 4-22.

[32] H. Moravec and A. Elfes, "High Resolution Maps From Wide Angle Sonar." in *Proc. IEEE Int. Conf. Robotics and Automation*, 1986, pp. 116-121.

[33] Polaroid, *Ultrasonic Ranging Sys* Cambridge, MA, 1982.

[34] E. Säckinger, B.E. Boser, J. Bromley, Y. LeCun and L.D. Jackel, "Application of the ANNA neural network chip to high-speed character recognition." *IEEE Transactions on Neural Networks*, pp 498-505, 1992.

[35] J. M. Zurada, *Introduction to Artificial Neural Systems*. West Publishing Company, St. Paul, MN, 1992.