Genetic Algorithm Optimization of a Convolutional Neural Network for Autonomous Crack Detection

Robert Ouellette
Open Thoughts Research
(Waseda University & GMD-JRL)
2-1-7-902 Komyo,Yahatanishi-ku
Kitakyushu, Japan 807-0824
Email: rpo@openthoughts.com

Matthew Browne
GMD-Japan Research Laboratory
Collaboration Center
2-1 Hibikino, Wakamatsu-ku
Kitakyushu, Japan 808-0135
Email: matthew.browne@gmd.gr.jp

Kotaro Hirasawa
Waseda University
Information, Production and Systems
2-7 Hibikino, Wakamatsu-ku
Kitakyushu, Japan 808-0135
Email: hirasawa@waseda.jp

***Abstract*— Detecting cracks is an important function in building, tunnel, and bridge structural analysis. Successful automation of crack detection can provide a uniform and timely means for preventing further damage to structures. This laboratory has successfully applied convolutional neural networks (CNNs) to online crack detection. CNNs represent an interesting method for adaptive image processing, and form a link between artificial neural networks and finite impulse response filters. As with most artificial neural networks, the CNN is susceptible to multiple local minima, thus complexity and time must be applied in order to avoid becoming trapped within the local minima. This paper employs a standard genetic algorithm (GA) to train the weights of a 4-5x5 filter CNN in order to pass through local minima. This technique resulted in a 92.3±1.4% average success rate using 25 GA-trained CNNs presented with 100 crack (320x240 pixel) images.**

I. Introduction

We wish to apply crack detection as one form of fault detection in communal sewer pipes. "'Crack' or 'no crack'" crack detection could easily be done with simple edge or line detectors. However, our application for crack detection dictates that the crack detector categorize the cracks within 3 main categories and 5 sub-categories. Thus a total of 15 different predefined subtypes of cracks must be correctly classified. Also, because we eventually need to classify other pipe faults (e.g. abnormal pipe bending, detection of foreign objects such as plant roots, and misaligned pipe joints), it is readily apparent that simple edge detection will not suffice. We need a pattern recognition technique that not only detects faults, but one that is also able to classify each fault into appropriate predefined categories.

Mimicking evolutionary processes found in nature, genetic algorithms are stochastic search methods used in optimization problems. This paper applies a standard GA to optimize the weights (4-5x5 filters) of a 2-layer CNN. The decision to use the GA for optimization rather than backpropagation or other traditional techniques was made because optimization of the network only needs to be done once and because we are concerned with accuracy much more so than optimization time. The GA does not guarantee the best results unless all possible genomes are applied to the problem, however the strength of the GA is that the GA is less susceptible to local minima.

This paper first describes our application of CNNs to crack detection and artificial image processing problems. Then we explain the methodology of applying GAs to modify the weights of the CNN and discuss the results. Lastly we make our concluding remarks and note enhancements that we believe would improve the system overall.

II. The Convolutional Neural Network

The term *convolutional neural network* (CNN) is used to describe an architecture for applying neural networks to two-dimensional arrays (usually images), based on spatially localized neural input. This architecture has also been described as the technique of shared weights or local receptive fields [1], [2], [3] and is the main feature of Fukushima's *neocognitron* [4], [5]. Le Cun and Bengio [6] note three architectural ideas common to CNNs: local receptive fields, shared weights (weight averaging), and often, spatial downsampling. Processing units with identical weight vectors and local receptive fields are arranged in an spatial array, creating an architecture with parallels to models of biological vision systems [6]. A CNN image mapping is characterized by the strong constraint of requiring that each neural connection implements the same local transformation at all spatial translations. This dramatically improves the ratio between the number of degrees of freedom in the system and number of cases, increasing the chances of generalization [7]. This advantage is significant in the field of image processing, since without the use of appropriate constraints, the high dimensionality of the input data generally leads to ill-posed problems. To some extent, CNNs reflect models of biological vision systems [8].

In the CNN architecture, the 'sharing' of weights over processing units reduces the number of free variables, increasing the generalization performance of the network. Because weights are replicated over the spatial array, leading to intrinsic insensitivity to translations of the input - an attractive feature for image classification applications. CNNs have been shown to be ideally suited for implementation in hardware, enabling very fast real-time implementation [9]. Although CNN have not been widely used in image processing, they have been applied to handwritten character recognition [2], [9], [10], [11] and face recognition [7], [8], [12].

The present paper presents an application of CNN to an applied problem in mobile robotics. In particular, the visual system of a KURT2 [13] autonomous mobile robot designed for sewer inspection and equipped with an infra-red video camera. The task of the the robot is the oon-linedetection of cracks and other faults in sewer pipes. The cracks are defined by a relatively distinctive space-frequency structure. However, they are often embedded in a variety of complex textures and other spurious features. Lighting and reflection effects present an additional source of difficulty. The implementation and performance of the CNN architecture is discussed in terms of this application. The CNN is also applied to certain artificial image processing problems.

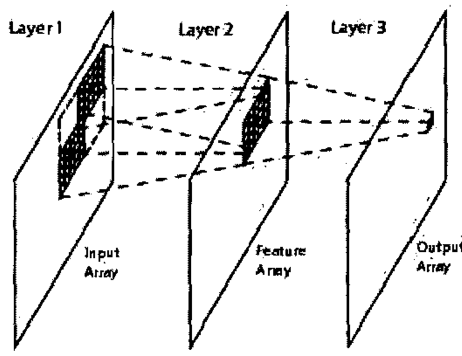

Fig. 1. Architecture of a CNN with a single convolutional neuron in two layers. A 5x5 filter at two different translations is shown mapping to shaded pixels in the first feature array. Shaded pixels in turn are part of the local feature information which is mapped to the output (or second feature) array by a second 5x5 filter.

Figure 1 shows the architecture of a CNN with two layers of convolution weights and one output processing layer. Neural weights in the convolution layers are arranged in an 2-D filter matrices, and convolved with the preceding array. In figure 1, a single layer 1 neural filter is shown operating at two different spatial translations on the input array. The output (shaded pixel) forms a spatially distributed feature map, which is processed by the second convolutional layer, and passed to the output array. Downsampling of the feature array may be implemented between the convolution layers. For fixed filter sizes, this has the effect of increasing the spatial range of subsequent layers, while reducing the level of spatial resolution. As with most neural networks, the exact architecture of a CNN should depend on the problem at hand. This involves determination of: the mixture of convolutional and downsampling layers (if any), filter-weight sizes, number of neurons, and interconnectivity between layers.

Define two-dimensional MxM filters $\mathbf{W}_{l,j,k}$ as the neural weights between the l^{th} and j^th feature arrays of layers $k-1$ and k, respectively. Feature arrays $\mathbf{F}_{j,k}$ are generated

$$\mathbf{F}_{j,k} \quad = \quad g\big(\sum_{l} \mathbf{W}_{l,j,k} \otimes \mathbf{F}_{l,k-1} + b_{j,k}\big) \qquad (1)$$

with $\otimes$ the 2-D convolution operator, $b_{j,k}$ the neural bias term, and g the transfer function (i.e. logsig). Note that $\mathbf{F}_{1,0}$ may be treated as the input array, and $\mathbf{F}_{1,K}$ as the output array. Regardless of the dimensions of the input array, the weight sharing property reduces the number of free network weight parameters in the k^{th} layer to M_k^2LJ. Figure 2 shows an example two layer CNN with multiple neurons in each layer, and an intermediate downsampling layer.

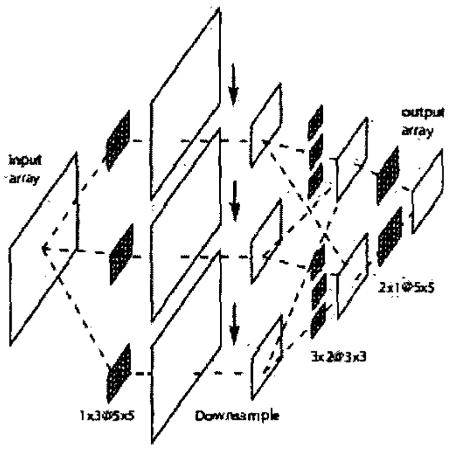

Fig. 2. Architecture of a fully interconnected CNN with multiple neurons in the convolution layer and an intermediate downsampling layer.

Although previous studies have treated the input to a CNN as a single array, it is clearly possible for the CNN architecture to simultaneously process multiple input arrays. This raises the interesting possibility of treating RGB or YUV color images as three concurrent input arrays, enabling the simultaneous learning of color and shape information. We also propose the idea of utilizing *separable* convolution kernels; that is, weight matrices $\mathbf{W}_{l,j,k}$ that may be formed by the outer product of a row and a column vector. This leads to the implementation of the row and column convolution operations separately

$$\mathbf{F}_{j,k} = g\big(\sum_{l} \mathbf{w}^{R}_{l,j,k} \otimes \big(\mathbf{w}^{C}_{l,j,k} \otimes \mathbf{F}_{l,k-1}\big) + b_{j,k}\big). \qquad (2)$$

This further reduces the degrees of freedom in each layer to $2M_kLJ$, and similarly speeds implementation of the filters. Of course, this also leads to a loss of flexibility in the types of features that may be extracted; in particular, orientation selectivity is lost. This may be useful in image processing, where there is sometimes an advantage in orientation independent feature detectors.

In the case of crack detection, we are trying to find a set of filters that we can run over an image such that the filters allow only cracks to pass through (a "crack-pass" filter, if you

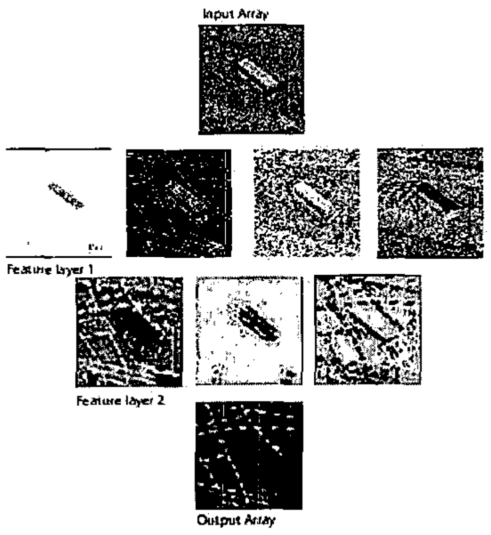

Fig. 3. Input, feature, and output arrays of a convolution network applied to detecting road markers.

will) to be classified at some later time. A common approach to training such filters is to use artificial neural networks (ANNs). Our method of applying the ANN to this problem takes advantage of the innate spatial features of the 2d image. As a result, the spatially-related neuron activations, through their activation, are actually performing a 2d convolution on the subject image.

III. GA Implementation

The GA we used to train the weights of the CNN is a "standard" GA. That is, the GA evolution takes the following steps:

1) Initialization of the population
2) Evaluating the fitness of each member
3) Selecting parents to produce offspring for the next generation
4) Crossing parents to produce offspring (i.e. "crossover")
5) Mutating the offspring by chance
6) Continuing evaluation, reproduction, and mutation until evaluation criteria has been reached

Integration of the GA with the CNN involves the following steps:

1) Initialization of each chromosome with random values
2) Substituting the weights of the CNN with the values of a selected chromosome
3) Simulating the CNN using the newly realized weights (this produces a output image)
4) Subtracting the resultant image from the target image to judge the fitness of the current member
5) Repeating the simulation for all members in the population
6) Selecting the parents of the next generation through a roulette process
7) Crossing the parents to make new children
8) Mutating the child with 1% chance of mutation
9) Repeating simulation, evaluation, reproduction, and mutation until evaluation criteria has been met

A. Population Initialization

The original CNN used in our crack detection algorithm used back propagation character (8 bit) inputs without normalization. In order to compare the success of our GA-based method with the back propagation method, we kept the inputs the same size. This complicates the evolution process by increasing the range through which to vary the weights. Initial values for the GA were selected randomly with values from 0 to 255. Bias values were randomly selected between -128 and 128 to allow for evolution in either direction.

B. Fitness Evaluation

The fitness for each member is determined by subtracting the resultant image from the target image. An exact match between the resultant image and the target image gives the highest possible fit.

C. Selection

Parent selection is performed through a roulette type process. The success of each member is taken from the evaluation process and is added to the running sum of the fitnesses for all the members. The "low chance value" for the member is set as the running sum value before addition of the member's fitness. The "high chance value" is calculated by adding the member's fitness to its low chance value. Thus, the chance for selection for each member becomes the member's success rate divided by the total success rate. Actual selection is done by randomly selecting a number between zero and the total success rate. The list of members is queried and the member whose low and high chance values straddle the selected number is selected to become a parent. The process is repeated for the second parent.

D. Crossover

A number between 0 and the chromosome length is randomly selected. This number represents the crossover point for the two parents. The part of the first parent chromosome that runs until the crossover point is spliced with the part of the second parent chromosome that includes, and runs after, the crossover point. The whole new generation is selected in this manner. Elitism is employed by directly replacing the worst performer with the best performer. This encourages progressive adaptation but also increases the chance of becoming temporarily trapped within local minima.

E. Mutation and Creep

During the crossover step, each chromosome is processed for mutation and "creep". For mutation, each chromosome is given a 1% chance of mutation and, if selected, the chromosome is changed randomly within +/- 1%. Similarly, the chromosome itself is given a 1% chance of "creep"ing. Creep, introduced by Davis [14], refers to the drift of the members of an array around their current value. If the chromosome is selected to creep, the chromosome is scaled by a common random factor between -1 and 1.

F. Evaluation Criteria

Evaluation of each member was done by simulating the CNN using the chromosome parts as weights in the CNN, then pixel-wise subtracting the output and target images. The error was calculated as the sum of the absolute value of each pixel subtraction between the output and target images. Taking the absolute value was done rather than using the root mean square because it was programmatically easier. Evolution is stopped after one of two criteria is met, that is, if the error rate falls below 1% or the generation reaches 300. Given that the relatively small filters must account for all cracks and filter any noise, a 1% error rate is unlikely to ever occur.

IV. Implementation

Each original input image was copied to create a target/evaluation image. The resultant target images were hand-coded with binary "crack" or "no crack" data. The crack areas were colored in with black (value of 0), and non-crack areas were colored in with white (value of 255). Evaluation of each member was done by simulating the CNN using the chromosome parts as weights in the CNN, then pixel-wise subtracting the input and target images. The absolute value of each pixel subtraction was done rather than use the least squares approach.

In reality, crack space appears significantly less frequently than non-crack space. Thus, a representative image would include few, if any, cracks. In order to speed up the learning rate of the CNN, we presented it with four unique sets of images, each set taken from a pool of still images captured from sewer inspection video. The images start small (20x20 pixels) and get larger through a progression of small (20x20 pixels), medium (40x40 pixels), medium-large (100x100 pixels), and large (320x240 pixels) images. Each small and medium image was "cut" out from larger images and was carefully chosen such that roughly half of each test image was filled with crack space. Medium-large and large images were trained with cracks present. Since the existence of a crack is evaluated at the pixel level and not the image level (for training), the larger images give sufficient "non-crack" data for evaluating the gambit from true-crack to false crack, and the corresponding "non-crack" parts.

The progressive training approach from small to large images allows the GA/CNN to optimize crack detection without having to first process through large amounts of "white" space.

V. Simulation Results

Shown below are sets of four images: input test image, output target image, the image for the which the GA that had the best-training fit, and the corresponding output image from the same CNN, but trained by backpropagation. Each set of four images correspond to each of the four categories (small size image, medium size image, medium-large size image,

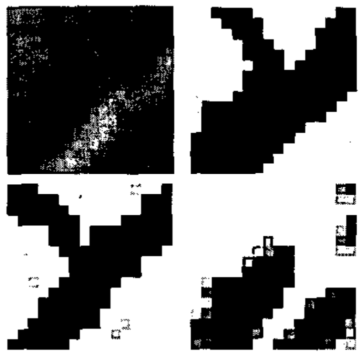

Fig. 4. From left to right: Original small (20x20 pixel) image. The hand coded target resized to 16x16 pixels for convolution comparison, and the resultant image after 500 generations.

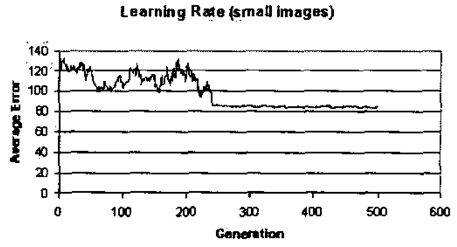

Fig. 5. The learning rate of the GA for the above small image (20x20 pixels) based on 500 generations.

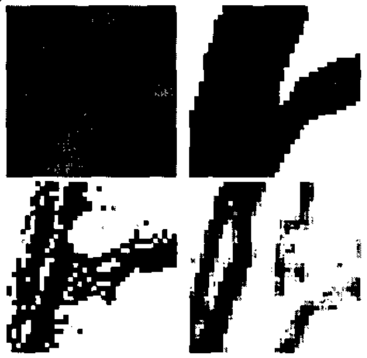

Fig. 6. Original medium (40x40 pixel) image. The hand coded target resized to 36x36 pixels for convolution comparison. The resultant image after 500 generations. The output of the CNN trained by backpropagation.

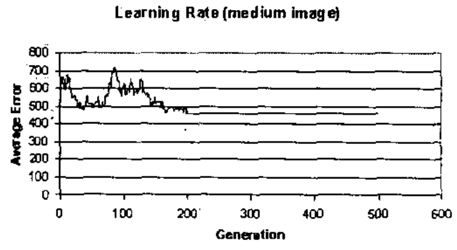

Fig. 7. The learning rate of the GA for the above medium-sized image (40x40 pixels) based on 500 generations.

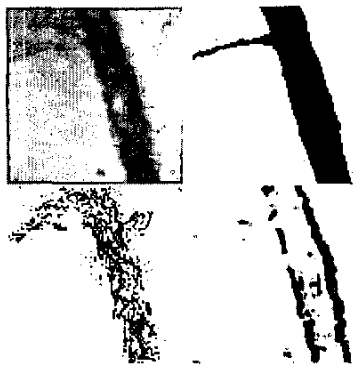

Fig. 8. Original medium-large (100x100 pixel) image. The hand coded target resized to 96x96 pixels for convolution comparison, and the resultant image after 500 generations. The output of the CNN trained by backpropagation.

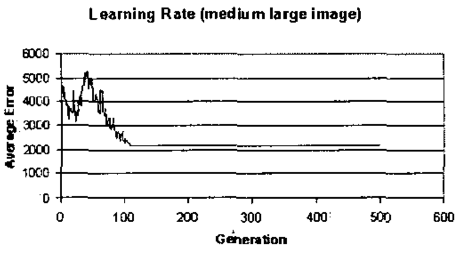

Fig. 9. The learning rate of the GA for the above medium-large image (100x100 pixels) based on 500 generations.

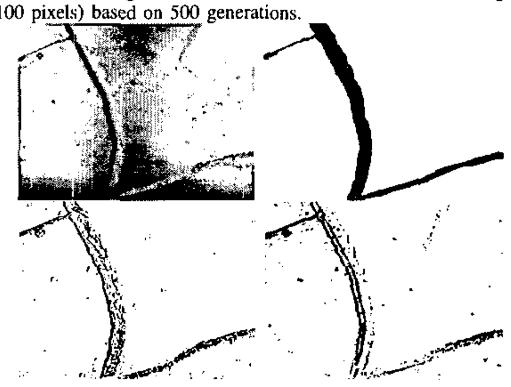

Fig. 10. Original large (320x240 pixel) image. The hand coded target resized to 316x236 pixels for convolution comparison, and the resultant image after 300 generations. The output of the CNN trained by backpropagation.

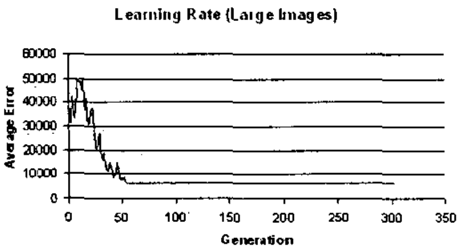

Fig. 11. The learning rate of the GA for above large image (320x240 pixels) based on 300 generations.

large image) after simulating through the appropriate number of generations. The images illustrate that while there is a fairly good match, the resultant image does not match perfectly with the target. The major factor affecting this difference is the size of the filters. Had the filters been the size of the image itself, then there could be a perfect match for each crack; however it is not computationally practical to apply such large filters in all possible combinations.

Below each set of pictures are graphs that show the average learning rate for each generation. The dips in the graphs illustrate the potential for local minima. A CNN trained using backpropagation could have become trapped in a local minima should the CNN have been unlucky to have gotten the "right" set of weights that would correspond to the dip as shown at around the 30th generation depicted in the learning rate graph shown in Figures 7 and 9.

In all, 25 GA-trained CNNs were presented with 100 crack (320x240 pixel) images. The overall crack detection rate for the GA-tuned CNN was 92.3±1.4%. In contrast, the backpropagation-trained CNN had an 89.8±3.2% success rate. Statistically, the standard GA implemented in tuning the CNN was no better than the backpropagation method. We refrained from employing a more advanced GA because while the results may become statistically important, without additional intelligence added to the system, the crack detector will not reach human-level detection.

VI. Conclusion

We have trained and simulated a convolutional neural network (CNN) using weights trained by standard genetic algorithm evolution methods and compared them with results from training the CNN through backpropagation. The GA approach resulted in a 92.3±1.4% accuracy for crack detection of 100 images. This was statistically no better than the backpropagation method, however in terms of training the CNN, using the GA was much simpler to implement. Intelligence needs to be added to the crack detection system because although more advanced GA techniques may potentially improve the error rate, they would not reach human-level of detection-our ultimate evaluation criteria. Further downsampling within the CNN and adding extra layers would likely improve the detection accuracy. Larger and more filters would also increase the detection accuracy however these methods, including further downsampling, are computationally expensive and should be avoided in order to perform fast, real-time crack detection. Further research will explore optimizing the size and number of filters as well as the amount of downsampling, all with respect to computation time and accuracy. Once a stable method of crack detection can be reached, classification of the different classes of cracks can begin.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representation by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds., vol. 1, pp. 318–362. Cambridge, MA: MIT Press, 1986.

[2] Y.B. Le Cun, J.S. Boser, D. Denker, R.E. Henderson, W. Howard, W. Hubbard, and L.D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 4, no. 1, pp. 541–551, 1988.

[3] K.J. Lang and G.E. Hinton, "Dimensionality reduction and prior knowledge in e-set recognition," in *Advances in Neural Information Processing Systems*, D.S. Touretzky, Ed., pp. 178–185. Morgan Kauffman, San Marteo, CA, 1990.

[4] K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: a neural model for a mechanism of visual pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, pp. 826–834, 1983.

[5] Kunihiko Fukushima, "Neocognitron: A hierachical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, no. 2, pp. 119–130, 1988.

[6] Y. Le Cun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, M.A. Arbib, Ed., pp. 255–258. MIT Press, Cambridge, MA, 1995.

[7] Steve Lawrence, C. Lee Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[8] B. Fasel, "Robust face analysis using convolutional neural networks," in *Proceedings of the International Conference on Pattern Recognition (ICPR 2002), Quebec, Canada*, 2002.

[9] E. Sackinger, B. Boser, J. Bromley, and Y. LeCun, "Application of the anna neural network chip to high-speed character recognition," *IEEE Transactions on Neural Networks*, vol. 3, pp. 498–505, 1992.

[10] Y. Le Cun, "Generalization and network design strategies," Tech. Rep. CRG-TR-89-4, Department of Computer Science, University of Toronto, 1989.

[11] Y. Bengio, Y. Le Cun, and D. Henderson, "Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and Hidden Markov Models," in *Advances in Neural Information Processing Systems*, Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, Eds. 1994, vol. 6, pp. 937–944, Morgan Kaufmann Publishers, Inc.

[12] Beat Fasel, "Facial expression analysis using shape and motion information extracted by convolutional neural networks," in *Proceedings of the International IEEE Workshop on Neural Networks for Signal Processing (NNSP 2002), Martigny, Switzerland*, 2002.

[13] F. Kirchner and J. Hertzberg, "A prototype study of an autonomous robot platform for sewerage system maintenance," *Autonomous Robots*, vol. 4, no. 4, pp. 319–331, 1997.

[14] L. Davis (ed), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.