Digital Signal Processing 21 (2011) 517–521

Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

A novel algorithm for all pairs shortest path problem based on matrix multiplication and pulse coupled neural network

Yudong Zhang *, Lenan Wu, Geng Wei, Shuihua Wang

School of Information Science and Engineering, Southeast University, China

ARTICLE INFO

Article history:
Available online 21 March 2011

Keywords:
All pairs shortest path
Pulse coupled neural network
Matrix multiplication
Parallel algorithm

ABSTRACT

All pairs shortest path (APSP) is a classical problem with diverse applications. Traditional algorithms are not suitable for real time applications, so it is necessary to investigate parallel algorithms. This paper presents an improved matrix multiplication method to solve the APSO problem. Afterwards, the pulse coupled neural network (PCNN) is employed to realize the parallel computation. The time complexity of our strategy is only $O(\log^2 n)$, where n stands for the number of nodes. It is the fastest parallel algorithm compared to traditional PCNN, MOPCNN, and MPCNN methods.

1. Introduction

Given a directed graph $G = \langle V, E \rangle$ and a length function $\omega : E \to R$, the "*all pairs shortest path*" problem, APSP in short, is to find the length of the shortest path between any two nodes (a length of a path is defined as the sum of its edge lengths). APSP is a classical problem with diverse applications today. For example, the vehicle routing problem (VRP) seeks to service a number of customers with a fleet of vehicles, which is an important problem in the fields of transportation, distribution and logistics [1]. Job shop scheduling (JSS) is an optimization problem in which ideal jobs are assigned to resources at particular times [2]. Traveling salesman problem (TSP) is a problem in combinatorial optimization in which a shortest tour is designed to visit each city exactly once [3]. Other applications include website page searching, abstract state machine, plant and facility layout, and VLSI design [4].

The APSP is an old problem. The first algorithm is due to Dijkstra [5]. Small improvements were made to this algorithm, mostly using sophisticated data structure [6]. All of them did not improve the $O(n^3)$ upper bound. Fredman's algorithm is $o(n^3)$, but its exponent is still 3 [7].

These traditional algorithms have major shortcomings: firstly, they are not suitable for networks with negative weights of the edges, i.e., in communication networks, the link weights represent the transmission line capacity and negative weights correspond to links with gain rather than loss. Secondly, the algorithms search only for the shortest route, but cannot determine any other closer-optimal solutions. Thirdly, they exhibit high computational complexity for real-time communications involving rapidly changing network topologies such as wireless ad hoc networks [8].

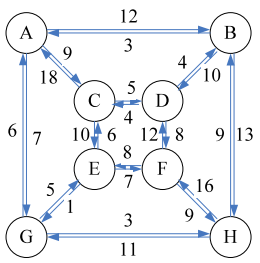
Hence, artificial neural network (ANN) has been applied for solving the APSP problem, since ANN's intrinsic property in solving large-scale problem parallel. The original work was done by Hopfield [9]. From then on, a great many of works has been done in the field of solving APSP with ANN [10]. However, there exist some major drawbacks in Hopfield networks and other ANN, which are listed as: 1) Invalidity of the obtained solutions. 2) Trial-and-error value process of the network parameters. 3) Low computation efficiency [11].

Pulse coupled neural network (PCNN) is a result of research on artificial neuron model that was capable of emulating the behavior of cortical neurons observed in the visual cortices of animal. According to the phenomena of synchronous pulse burst in the cat visual cortex, Eckhorn developed the linking field network. Since it does not need to pre-train, and inherits the advantages of ANN, PCNN has been used for various applications, such as image denoising [12], image thinning [13], image fusion [14], image segmentation [15], and pattern recognition [16], etc.

In Caulfield's work [17], PCNN is structured in such a way, that each point in the geometric maze figure corresponds to a neuron in the network, and the autowave in the PCNN travels from each neuron to its neighborhood neuron(s) along the maze from iteration to iteration of the network. Hence, it requires a great many of PCNN neurons to present only one edge. Qu [18] simplified the structure proposed by Caulfield, and advanced a Multi-Output PCNN model (MOPCNN) where each neuron in the network correspond to a node in the graph, namely, only one neuron to present an edge. However, the time consummation of each neuron is too large to practically application, and the algorithm is not quite suitable for continuous SP. Wang [19] proposed a modified PCNN (MPCNN), and applied the MPCNN to shortest path, but the algorithm needs large memory storage.

* Corresponding author.
E-mail address: zhangyudongnuaa@gmail.com (Y. Zhang).

1051-2004/$ – see front matter
doi:10.1016/j.dsp.2011.02.004



(a)

$$\begin{bmatrix} 0 & 12 & 9 & \text{inf} & \text{inf} & \text{inf} & 7 & \text{inf} \\ 3 & 0 & \text{inf} & 10 & \text{inf} & \text{inf} & \text{inf} & 13 \\ 18 & \text{inf} & 0 & 5 & 10 & \text{inf} & \text{inf} & \text{inf} \\ \text{inf} & 4 & 4 & 0 & \text{inf} & 8 & \text{inf} & \text{inf} \\ \text{inf} & \text{inf} & 6 & \text{inf} & 0 & 7 & 1 & \text{inf} \\ \text{inf} & \text{inf} & \text{inf} & 12 & 8 & 0 & \text{inf} & 9 \\ 6 & \text{inf} & \text{inf} & \text{inf} & 5 & \text{inf} & 0 & 3 \\ \text{inf} & 9 & \text{inf} & \text{inf} & \text{inf} & 16 & 11 & 0 \end{bmatrix}$$

(b)

Fig. 1. A small directed graph and its one-step distance matrix: (a) The graph; (b) One-step distance matrix.

This study also employed the PCNN, but the basic idea is total different. We proposed a novel parallel algorithm to solve APSP by matrix multiplication, and then PCNN was employed to simulate the matrix multiplication method. The strategy can be done within $O(\log^2 n)$, where n represents the number of nodes.

The structure of rest is organized as follows: Section 2 presents the basic concept of matrix multiplication was used for solving APSP problems, and proposed two acceleration strategies for improvement. Section 3 discusses how to use PCNN to simulate the matrix multiplication method. Besides, a simplified PCNN was proposed. Section 4 gives the experimental results, and compares our proposed method to PCNN, MOPCNN, and MPCNN. The results show the time complexity of our algorithm is the least. Final Section 5 is devoted to conclusion.

2. Matrix multiplication for APSP

Matrix multiplication is a classical method for solving the APSP problem. It is not eye-catching since it does not suit well for serial compute model. However, we find it efficient in the parallel neural network [20].

2.1. Traditional method

Suppose R represents the one-step distance matrix. R_{ij}, that lies at row i and column j, stands for the minimum distance from node i to node j within no more than one step. Fig. 1(a) shows a small directed graph, and Fig. 1(b) shows the corresponding one-step distance matrix.

Define a new matrix multiplication as follows. Suppose "+" as the minimum operation, "×" as addition. Then

$$R_{ij}^2 = (R \bullet R)_{ij} = \sum_k R_{ik} \times R_{kj} = \min_k (R_{ik} + R_{kj}) \tag{1}$$

It is obvious that $R_{ii} = 0$ $(\forall i)$. Hence, $R^2 = R \bullet R$ stands for the two-steps distance matrix, and R_{ij}^2 represents the minimum distance from node i to node j within no more than two steps. This is the same as $R^3, R^4, \ldots, R^n$, where n represents the number of nodes in the graph. It is obvious that R^n is the solution for APSP problem, since the nodes covered by the shortest path is no more than n [21].

2.2. Improvement I: Square method

If we want to attain R^n, it needs $n-1$ times of matrix multiplication. Can it be accelerated?

Lemma 1. *Our defined matrix multiplication still obeys the associative law.*

Proof. Suppose there are 3 matrices: A, B, and C. The number of columns of A is equal to that of rows of B. The number of columns of B is equal to that of rows of C. It is obvious that

Index	*Cost*
(1, 2, 3, 4, 5, 6, 7, 8)	(11, 9, 3, 4, 10, 8, 6, 2)
(2, 3, 6, 8)	(9, 3, 8, 2)
(3, 8)	(3, 2)
(8)	(2)

Fig. 2. A simple example of minimum selection.

$$\begin{aligned} \big[(AB)C\big]_{ij} &= \min_n \big[(AB)_{in} + C_{nj}\big] = \min_n \Big[\min_m (A_{im} + B_{mn}) + C_{nj}\Big] \\ &= \min_m \Big[\min_n (A_{im} + B_{mn} + C_{nj})\Big] \end{aligned} \tag{2}$$

$$\begin{aligned} \big[A(BC)\big]_{ij} &= \min_m \big[A_{im} + (BC)_{mj}\big] = \min_m \Big[A_{im} + \min_n (B_{mn} + C_{nj})\Big] \\ &= \min_m \Big[\min_n (A_{im} + B_{mn} + C_{nj})\Big] \end{aligned} \tag{3}$$

Hence, $(AB)C = A(BC)$. □

Then, the times of matrix multiplication for attaining R^n can be reduced to only $\log(n)$. This idea—square method—is depicted as follows: Firstly, we construct the one-step distance matrix R. Then, we attain R^2 by $R \times R$, attain R^4 by $R^2 \times R^2$, attain R^8 by $R^4 \times R^4$. Repeat this process until attaining R^n. If n is not a power of 2, then suppose m is the minimum integer that is large than n and is a power of 2. We attain R^m instead of R^n by aforementioned processing.

2.3. Improvement II: Minimum selection

It takes $O(n)$ time to select the minimum from a list of nonnegative numbers. However, it can be accelerated by a basic technique. For simplicity n is assumed as a power of 2. Suppose that the list is $c(1), c(2), \ldots,$ we go through a tournament under the rule that if $c(i) < c(i+1)$ for an odd i, then i is chosen as a winner for the next stage, or $i+1$ otherwise. It reduces the $O(n)$ time to only $\log(n)$ time via this technique. Fig. 2 shows a simple example for $n = 8$.

3. Parallel computation by PCNN

Although the improved matrix multiplication method can hasten the procedures greatly, it still needs large time and memory resource. Here we discuss how to use PCNN to realize the strategy in a parallel way.

3.1. Introduction to PCNN

A typical neuron of PCNN consists of 3 parts: the receptive fields, the modulation fields, and the pulse generator. It is shown in Fig. 3.

Suppose N is the total number of iterations and n is current iteration, the neuromime of PCNN can be described by the following equations.

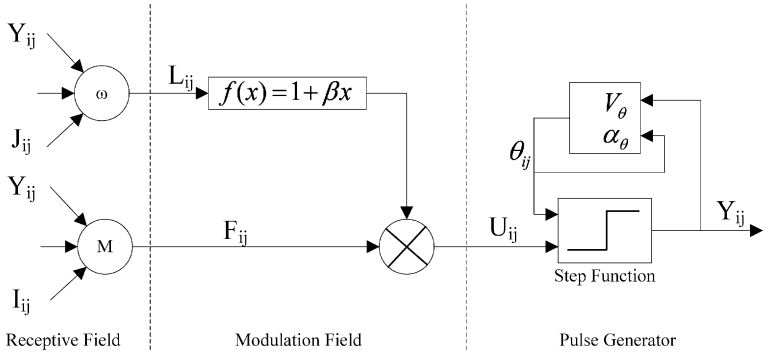


Fig. 3. Neuromime of PCNN.

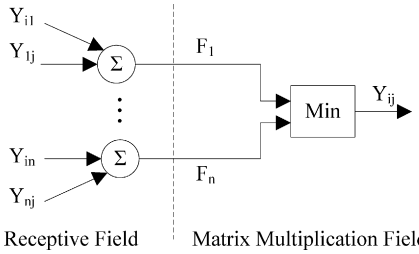


Fig. 4. The novel structure of a neuron for matrix multiplication.

$$F_{ij}[n] = \exp(-\alpha_F)F_{ij}[n-1] + V_F \sum m_{ijkl} Y_{kl}[n-1] + I_{ij} \tag{4}$$

$$L_{ij}[n] = \exp(-\alpha_L)L_{ij}[n-1] + V_L \sum \omega_{ijkl} Y_{kl}[n-1] \tag{5}$$

$$U_{ij}[n] = F_{ij}[n]\big(1 + \beta L_{ij}[n]\big) \tag{6}$$

$$Y_{ij}[n] = \begin{cases} 1, & U_{ij}[n] > \theta_{ij}[n-1] \\ 0, & U_{ij}[n] \leqslant \theta_{ij}[n-1] \end{cases} \tag{7}$$

$$\theta_{ij}[n] = \exp(-\alpha_\theta)\theta_{ij}[n-1] + V_\theta Y_{ij}[n-1] \tag{8}$$

Here the (i, j) pairs present the position of neuron. F, L, U, Y, and θ are feeding inputs, linking inputs, internal activity, pulse output, and dynamic threshold, respectively. α_F, α_L and α_θ are time constants for feeding, linking and dynamic threshold. V_F, V_L and V_θ are normalizing constants, M and ω are the synaptic weights, and I_{ij} and J_{ij} are external inputs. β is the strength of the linking.

The whole working process is as follows: if there is a pulse fired, then the threshold θ will increase such that there will be no output during the next step. Therefore the threshold will exponentially decrease until it becomes smaller than the inner activity U, so that there will be another pulse fired. The above processes cycle all the time.

3.2. Simplified PCNN

In order to realize matrix multiplication based on PCNN, traditional PCNN was simplified. Fig. 4 shows the novel structure of the simplified neuron.

The n by n neurons are disposed in the grids of a 2D plane. The neuron lying at row i, column j, and the kth cycle, stands for the shortest path from node i to node j within no more than 2^k steps. The initialization of PCNN is done by setting the values of each neuron as the corresponding value of one-step distance matrix. Then after each cycle, the corresponding matrix is the square of itself at last cycle. Since it only takes $\log(n)$ time for matrix multiplication, and $\log(n)$ time to attain the minimum from a specified list, the computational complexity of our proposed algorithm based on PCNN and matrix multiplication is only $O(\log^2 n)$.

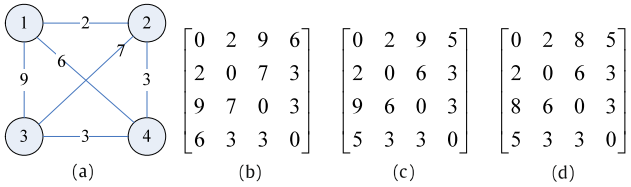


Fig. 5. A small symmetric network: (a) Graph; (b) R; (c) R^2; (d) R^4.

4. Experiments

To verify the theoretical results and the effectiveness of the proposed algorithm, several experiments have been carried out on the platform of P4 IBM with 3 GHz main frequency and 2 GB memory, running under Windows XP operating system. The algorithm was developed via Matlab 2010a.

4.1. A symmetric graph

The first is based on a symmetric weighted graph with 4 nodes and 5 edges, as shown in Fig. 5(a). It takes only 2 steps to obtain the final result. Figs. 5(b–d) show the results of each step. Fig. 5(b) is the original one-step distance matrix generated by PCNN initialization. Fig. 5(c) is the results of the first cycle of PCNN. Fig. 5(d) is the results of the second cycle.

Let us trace the change of neuron at row 2 and column 3. It corresponds to the distance from node 2 to node 3 in Fig. 5. Its value is 7 at the initialization stage. After the first cycle, its value decreased to 6, since PCNN has found a two-step route: 2-4-3. Another example is the neuron at row 1 column 3. Its value is 9 at the initialization stage. However, after 2 cycles, its value drops to 8, since PCNN has found a three-step route: 1-2-4-3.

4.2. An asymmetric graph

The second experiment is to find the shortest path of an asymmetric graph, which is shown in Fig. 1. There are 8 nodes and 24 edges in the graph. The results of each cycle are shown in Fig. 6. Only three steps can get the shortest path between any two nodes in the graph.

4.3. Time comparison

Suppose there are n nodes in the graph and the furthest length from one node to another node is S—usually $n^2 \ll S$, Table 1 shows the differences of our algorithm and other PCNN based algorithms, indicating that the proposed method costs more space than MOPCNN method but much less space than both PCNN and MPCNN method. Moreover, the time complexity of the proposed algorithm is the most least.

$$\begin{bmatrix} 0 & 12 & 9 & \inf & \inf & \inf & 7 & \inf \\ 3 & 0 & \inf & 10 & \inf & \inf & \inf & 13 \\ 18 & \inf & 0 & 5 & 10 & \inf & \inf & \inf \\ \inf & 4 & 4 & 0 & \inf & 8 & \inf & \inf \\ \inf & \inf & 6 & \inf & 0 & 7 & 1 & \inf \\ \inf & \inf & \inf & 12 & 8 & 0 & \inf & 9 \\ 6 & \inf & \inf & \inf & 5 & \inf & 0 & 3 \\ \inf & 9 & \inf & \inf & \inf & 16 & 11 & 0 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 0 & 12 & 9 & 14 & 12 & \inf & 7 & 10 \\ 3 & 0 & 12 & 10 & \inf & 18 & 10 & 13 \\ 18 & 9 & 0 & 5 & 10 & 13 & 11 & \inf \\ 7 & 4 & 4 & 0 & 14 & 8 & \inf & 17 \\ 7 & \inf & 6 & 11 & 0 & 7 & 1 & 4 \\ \inf & 16 & 14 & 12 & 8 & 0 & 9 & 9 \\ 6 & 12 & 11 & \inf & 5 & 12 & 0 & 3 \\ 12 & 9 & \inf & 19 & 16 & 16 & 11 & 0 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 0 & 12 & 9 & 14 & 12 & \inf & 7 & 10 \\ 3 & 0 & 12 & 10 & \inf & 18 & 10 & 13 \\ 18 & 9 & 0 & 5 & 10 & 13 & 11 & \inf \\ 7 & 4 & 4 & 0 & 14 & 8 & \inf & 17 \\ 7 & \inf & 6 & 11 & 0 & 7 & 1 & 4 \\ \inf & 16 & 14 & 12 & 8 & 0 & 9 & 9 \\ 6 & 12 & 11 & \inf & 5 & 12 & 0 & 3 \\ 12 & 9 & \inf & 19 & 16 & 16 & 11 & 0 \end{bmatrix}$$

(c)

$$\begin{bmatrix} 0 & 12 & 9 & 14 & 12 & \inf & 7 & 10 \\ 3 & 0 & 12 & 10 & \inf & 18 & 10 & 13 \\ 18 & 9 & 0 & 5 & 10 & 13 & 11 & \inf \\ 7 & 4 & 4 & 0 & 14 & 8 & \inf & 17 \\ 7 & \inf & 6 & 11 & 0 & 7 & 1 & 4 \\ \inf & 16 & 14 & 12 & 8 & 0 & 9 & 9 \\ 6 & 12 & 11 & \inf & 5 & 12 & 0 & 3 \\ 12 & 9 & \inf & 19 & 16 & 16 & 11 & 0 \end{bmatrix}$$

(d)

Fig. 6. Results of each step of Fig. 1: (a) R; (b) R^2; (c) R^4; (d) R^8.

Table 1
Comparison of our proposed algorithm and others.

Algorithm	No. of neurons	Time complexity
PCNN [17]	S	$O(S^2)$
MOPCNN [18]	n	$O(n \times S)$
MPCNN [19]	S	$O(S^2)$
Proposed method	n^2	$O(\log^2 n)$

5. Conclusions

In this study, we presented an improved algorithm based on matrix multiplication to solve APSP problem. Besides, we discuss the realization by a simplified PCNN. The computational complexity of our proposed algorithm is only $O(\log^2 n)$ for parallel computational model. It is a deterministic method which would guarantee the globally solutions. The shortest paths of all pairs are attained by running the network only once. The highly parallel computation of our proposed algorithm can be realized with VLSI. It is easy to extend this algorithm to other cases.

Acknowledgments

This research was supported under the following projects: 1) National Natural Science Fund (60872075); 2) National High Technology Development Plan (2008AA01Z227); 3) Senior University Technology Innovation Essential Project Cultivation Fund (706028).

References

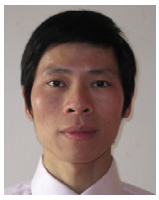
[1] K. Ghoseiri, S.F. Ghannadpour, A hybrid genetic algorithm for multi-depot homogeneous locomotive assignment with time windows, Applied Soft Computing 10 (1) (2010) 53–65.
[2] S.E. Kesen, S.K. Das, Z. Güngör, A genetic algorithm based heuristic for scheduling of virtual manufacturing cells (VMCs), Computers & Operations Research 37 (6) (2010) 1148–1156.
[3] Y. Zhang, et al., Find multi-objective paths in stochastic networks via chaotic immune PSO, Expert Systems with Applications 37 (2010) 1911–1919.
[4] S. Chaudhury, K.T. Sistla, S. Chattopadhyay, Genetic algorithm-based FSM synthesis with area-power trade-offs, Integration, the VLSI Journal 42 (3) (2009) 376–384.
[5] S. Peyer, D. Rautenbach, J. Vygen, A generalization of Dijkstra's shortest path algorithm with applications to VLSI routing, Journal of Discrete Algorithms 7 (4) (2009) 377–390.
[6] M.G. Sánchez-Torrubia, C. Torres-Blanc, M.A. López-Martínez, PathFinder: a visualization eMathTeacher for actively learning Dijkstra's algorithm, Electronic Notes in Theoretical Computer Science 224 (2009) 151–158.
[7] H. Gibbons, Definite descriptions and Dijkstra's odd powers of odd integers problem, Electronic Notes in Theoretical Computer Science 225 (2009) 83–98.
[8] P. Ferragina, I. Nitto, R. Venturini, On compact representations of all-pairs-shortest-path-distance matrices, Theoretical Computer Science 411 (34–36) (2010) 3293–3300.
[9] G. Pajares, M. Guijarro, A. Ribeiro, A Hopfield neural network for combining classifiers applied to textured images, Neural Networks 23 (1) (2010) 144–153.
[10] S. Effati, M. Jafarzadeh, Nonlinear neural networks for solving the shortest path problem, Applied Mathematics and Computation 189 (1) (2007) 567–574.
[11] D. Zhou, R. Nie, D. Zhao, Analysis of autowave characteristics for competitive pulse coupled neural network and its application, Neurocomputing 72 (10–12) (2009) 2331–2336.
[12] Y. Zhang, L. Wu, Improved image filter based on SPCNN, Science in China E Edition: Information Science 51 (12) (2008) 2115–2125.
[13] X. Gu, D. Yu, L. Zhang, Image thinning using pulse coupled neural network, Pattern Recognition Letters 25 (9) (2004) 1075–1084.
[14] M. Li, W. Cai, Z. Tan, A region-based multi-sensor image fusion scheme using pulse-coupled neural network, Pattern Recognition Letters 27 (16) (2006) 1948–1956.
[15] Y. Zhang, L. Wu, Segment-based coding of color images, Science in China F Edition: Information Science 52 (6) (2009) 914–925.
[16] Y. Zhang, L. Wu, Pattern recognition via PCNN and Tsallis entropy, Sensors 8 (11) (2008) 7518–7529.
[17] H.J. Caulfield, J.M. Kinser, Finding the shortest path in the shortest time using PCNN's, IEEE Transactions on Neural Networks 10 (3) (1999) 604–606.
[18] H. Qu, Z. Yi, A new algorithm for finding the shortest paths using PCNNs, Chaos, Solitons & Fractals 33 (4) (2007) 1220–1229.
[19] X. Wang, H. Qu, Z. Yi, A modified pulse coupled neural network for shortest-path problem, Neurocomputing 72 (13–15) (2009) 3028–3033.
[20] M. Krotkiewski, M. Dabrowski, Parallel symmetric sparse matrix–vector product on scalar multi-core CPUs, Parallel Computing 36 (4) (2010) 181–198.
[21] J.F. Sibeyn, External matrix multiplication and all-pairs shortest path, Information Processing Letters 91 (2) (2004) 99–106.

Yudong Zhang was born in Suzhou, China. He received a B.S. degree and M.S. in electronic engineering from Nanjing University of Aeronautics and Astronautics, Nanjing, in 2004 and 2007 respectively. He received a Ph.D. degree in communication and signal processing from Southeast University, Nanjing, in 2010.

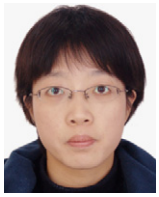
Now he works as a post-doctor in Columbia University and a research scientist in New York State Psychology Institute. His research interests in computer science are in the

areas of data mining for artificial neural network, decision tree, expert system, artificial intelligence, genetic algorithm, and particle swarm optimization. He is the first author of 15 papers published in SCI-index journals and 35 papers published in EI-indexed journals or conferences. He is an active consultant for industry in these areas.

Lenan Wu was born in Quanzhou, China. He received a B.S. degree and M.S. in electronic engineering from Nanjing University of Aeronautics and Astronautics Nanjing, in 1982 and 1987 respectively. He received a Ph.D. degree in signal and information processing from Southeast University, Nanjing, in 1997. Now he works as a professor post-doctor in Southeast University. His research interests include multimedia information processing and communication signal processing. He has published over 290 journal papers and is the author or co-author of nine books.



Geng Wei received the B.S. and M.S. degrees from the Wuhan University of China in 1999 and 2002, respectively, and the Ph.D. degree in signal and information processing from Huazhong University of Science & Technology in 2007. His current research interests include image and video processing, object identification and tracking, and multimedia communication.



Shuihua Wang was born in Qidong, China. She received a B.S. degree in the School of Information and Science Engineering, Southeast University, Nanjing, in 2008. Now she researches as a master candidate in Grove School of Engineering, City College of New York. Her research interests in electronic engineering involve image processing, wavelet analysis, and artificial neural network. She is the author of 5 papers published in SCI-indexed journals and 14 papers published in EI-indexed journals and conferences.