유전자 알고리즘을 이용한 물체인식을 위한 특징점 일치에 관한 연구

이 진 호† · 박 상 호††

요 약

모델을 이용한 물체인식을 모델영상들과 입력영상 간의 그래프 매칭과정으로 정의하였다. 본 논문에서는 그래프 매칭 문제를 최적화문제로 모델링 하였고 최적화문제 해결을 위하여 유전자 알고리즘을 제안하였다. 이를 위하여 적합성함수, 자료구조, 유전연산자들이 개발되었다. 제안된 유전자 알고리즘이 이차원 영상에서 부분적으로 겹쳐진 물체들을 인식하기 위한 모델영상과 입력영상 간의 특징점들을 일치시킴을 시뮬레이션을 통하여 보였다. 제안된 방법의 성능을 신경회로망을 이용한 방법과 비교하였다.

A Study on Feature Points Matching for Object Recognition Using Genetic Algorithm

Jin-Ho Lee† · Sang-Ho Park††

ABSTRACT

The model-based object recognition is defined as a graph matching process between model images and an input image. In this paper, a graph matching problem is modeled as an optimization problem and a genetic algorithm is proposed to solve the problem. For this work, fitness function, data structure, and genetic operators are developed. The simulation results are shown that the proposed genetic algorithm can match feature points between model image and input image for recognition of partially occluded two-dimensional objects. The performance of the proposed technique is compared with that of a neural network technique.

1. Introduction

The model-based object recognition task can be defined as graph matching process between model graph and input graph. The graph matching approach has been previously used for object recognition [1], stereo vision [2], edge detection [3], and motion detection [4]. Almost all the matching algorithms try to find a partial correspondence between the model and the scene features assuming unmatched features are hidden or distorted. The major difficulty with graph matching based object recognition is how to solve the combinatorial time complexity problem in searching of an optimal matching solution. In this paper, a feature points matching technique for the model-based object recognition is proposed. A feature graph for model image and a

† 정 회 원 : 경일대학교 공과대학 컴퓨터공학과 교수
†† 정 회 원 : 안동대학교 정보통신공학과 교수
논문접수 : 1998년 10월 7일, 심사완료 : 1999년 2월 2일

feature graph for input image are constructed using high curvature points in objects. The graph matching problem is modeled as an optimization problem and a genetic algorithm is implemented to solve the problem. Genetic algorithms [5] are randomized global search algorithm by maintaining a population of potential solutions. The strength of genetic algorithms lies in finding good optimal solutions very quickly in a complex search space. This is the reason of using genetic algorithms for graph matching based object recognition.

Various techniques have been suggested for model-based object recognition. In [6], objects are approximated by polygons and the points of high curvatures (corners) are used for hypotheses generation. A graph is constructed between the model corners and scene corners in order to find the mutually compatible corners. Extraction of the largest set of mutually compatible matches from the graph forms a model hypothesis. The hypothesis generation algorithm finds the largest part of the model boundary consistent with the scene. Ayache [7] has used polygon line segments as features. The longer line segments determine 'priviledged' features. These privileged segments determine initial hypotheses when matched to scene line segments. Each hypothesis receives a quality score. The quality score determines how much the model resembles the scene object. Models with quality scores above a threshold are said to be possible objects present in the scene. More recent technique [1] used Hopfield neural networks for object recognition. The two-dimensional model-based object recognition is expressed in terms of excitatory and inhibitory connections between neurons. A model graph and a scene graph is constructed from feature points. The object recognition task is formulated as graph matching process. They have shown that it is possible to map the graph matching problem onto a Hopfield neural network with an appropriate energy function.

In our approach, the object recognition algorithm is defined as a graph matching process, and that is modeled as a maximization of fitness function. For each two-dimensional object, a model graph is constructed from its feature primitives where each node in the graph represents a feature point. Each feature point is connected to other nodes by an arc representing the relationship or compatibility between them. All model graphs are then integrated into a model database to form what is called a global model graph. For the recognition of objects in the input scene, an input graph is constructed using the same method which is then matched against the global model graph to identify and locate the presence of the models in the input scene. High curvature points (corners) are used in our algorithm as feature points. To recognize objects in the input scene, a graph matching process between the global model graph and input graph is performed. The graph matching process is formulated as an optimization problem, and that is implemented by genetic algorithms to find the optimal solution. A fitness function for the feature points matching is derived which represents the constraints that the nodes of the two graphs should satisfy in order to find the best matches. Appropriate genetic operators and representation structure for a population are also developed. Experimental results are presented in terms of matching rates and those are compared with the results of a Hopfield neural network based graph matching technique [1].

2. Feature Extraction and Model Building

The features that are usually used in a matching process are the global features or the local features. The commonly used global features are the moments of inertia, Hough transform [8], centroid, area, and perimeter. Global features are very useful for recognition of single isolated objects, but their performance is poor in the case of partially occluded objects. Local features are more appropriate for recognition of occluded objects since most objects can be recognized with a subset of their local

features. Local features are extracted from the boundary of the object or from a localized region of the object. Local features that have been used for object recognition are corners [6], lines [7], and arcs [9]. High curvature points are used in our algorithm as local features.

Real images of keys are digitized into 8-bit gray level images. Each gray level image is converted into a binary image using the optimal thresholding method [10] which is based on the discriminant criterion. An averaging filter is applied before thresholding in order to suppress the noise. Each binary image is then scanned vertically and horizontally to extract the boundaries of the objects as a closed contour. A chain code is constructed using the object's boundary. This boundary is segmented into straight lines by a polygonal approximation algorithm [11].

To create the global model graph we construct a model graph from each prototype model using the dominant points (corners) of the boundary as the nodes of the model graph. Each node in the graph has its own local feature property as well as relational properties with other nodes. The local feature property of the node is represented by the angle of the corresponding corner and its relational properties (the global information) by the distances between all other nodes in the graph. The global model graph is then constructed by integrating all the model graphs with appropriate interactions. By a similar procedure, an input graph is constructed for the input image which may consist of several overlapping objects. Object recognition task is formulated by matching the feature points between the global model graph and input graph. Interactions between the matched feature points in the global model graph and input graph is represented through a fitness function.

3. Genetic Algorithms for Graph Matching

3.1 Introduction to genetic algorithm

A genetic algorithm is iterative procedure that maintains a population during iterations and can find the optimal solution for a particular problem by seeking the maximum/minimum of the appropriate fitness function. A population consists of a number of strings which represent possible candidate solutions. At each iteration a new population is created from the previous population using a set of genetic operators. The basic procedure of a typical genetic algorithm is depicted in Fig. 1.

During each iteration t, called generation, strings in the current population $P(t)$ are evaluated on the basis of their values from the fitness function and have probability of selection for next generation. This iterative process of selecting new strings is called reproduction. To generate a new population for the next generation, usually two selected strings are recombined by specific genetic operators such as crossover and mutation. This procedure would continuously generate new populations until a termination condition is reached. After termination of the iteration, the best string in the final population is chosen as the solution. A genetic algorithm may be terminated by determining the maximum number of iteration or after finding an acceptable solution.

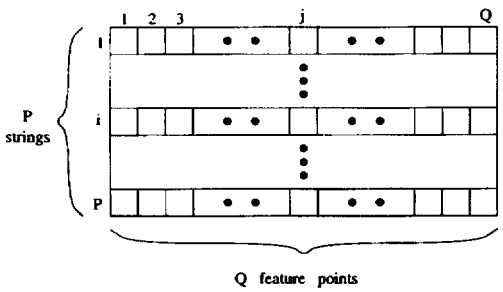
```
Set initial iteration t = 0
Initialize P(t)
Evaluate P(t)
while (termination condition not satisfied) do
begin
        Generate P(t+1) from P(t)
        Recombine P(t+1)
        Evaluate P(t+1)
        t = t+1
end
```

(Fig. 1) Procedure of genetic algorithms

3.2 The structure of strings

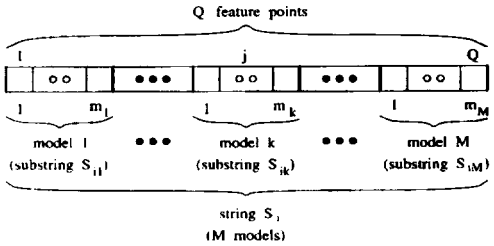
There are P strings in the population and each string consists of Q feature points representing all the features in the M models. A population is constructed by a two-dimensional array of size $P \times Q$ as shown in Fig. 2. A string S_i in the popu-

lation is composed of M substrings and substring S_{ik} has m_k feature points. Roughly speaking, the strings of artificial genetic systems are analogous to chromosomes in biological systems. Chromosomes are composed of genes which may take some values and strings in our system composed of substrings.

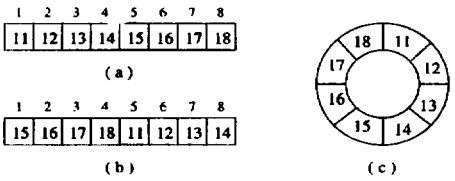


(Fig. 2) The structure of a population

Each substring S_{ik} represents the result of matching the feature points between the input graph and the *kth* model graph. The structure of a string is shown in Fig. 3. The value $S_{ik}(l)$ at the *lth* position in the substring S_{ik} represents a feature point in the input graph that matches the *lth* feature point in the *kth* model graph. If there is no feature point in the input graph, this feature location is marked by an 'x' in our notation. For examples, $S_{i2}(1)$ = 16 means that the *16th* feature point in the input scene graph is matched with the first feature point in the second model graph and $S_{i2}(2)$ = x means that the second feature point in the second model graph has no matching candidate in the input graph. Each substring is structures to be like a ring which has neither beginning nor end point. Feature candidates from the input scene graph are always assigned in an increasing (or decreasing) order for each substring because the scene boundaries are always closed contours and the order of the feature points is preserved even though the objects could be in any orientation. The structure of a substring is depicted in Fig. 4. Another constraint for assigning a matched point from the input graph onto a S_{ik} is that the same input feature point cannot be assigned to the substring more than once.



(Fig. 3) The structure of a substring; a string is composed of M substrings



(Fig. 4) The structure of a substring as a ring;
(a) is an example of the assignment of feature point number to a substring S_{ik},
(b) is the new substring when a substring S_{ik} is rotated, and
(c) represents the structure of each substring as a ring

3.3 Fitness function

The fitness function, which estimates the goodness of the string, consists of the probability of selecting a string among strings in the population, and the probability of selecting a particular substring among substrings in the string. The goodness function or objective function for a substring(OFSS) is defined by

$$f_{ik} = \sum_{p=1}^{m_k} \sum_{q=1}^{m_k} V_{ikp} V_{ikq} D_{ikpq} \qquad (1)$$

where V_{ikp} represents the state of a possible match which takes a value of 1 when the *pth* feature point in the *kth* model graph matches with the feature point $S_{ik}(p)$ in the input graph. The decision of assigning a match between feature points in the model graph and in the input graph is determined

by the differences between the angles of two feature points. If this difference is less than or equal to a predefined threshold then these two points are considered as matching points. The compatibility measure D_{ikpq} takes value 1 when the *pth* and *qth* feature points in the *kth* model graph are matched with feature points $S_{ik}(p)$ and $S_{ik}(q)$, respectively, and the difference between the distance from $S_{ik}(p)$ to $S_{ik}(q)$ is less than or equal to a predefined threshold. The objective function of a string(OFS) is equivalent to the sum of the OFSS which is given by:

$$f_i = \sum_{k=1}^{M} f_{ik} = \sum_{k=1}^{M} \sum_{p=1}^{m_k} \sum_{q=1}^{m_k} V_{ikp} V_{ikq} D_{ikpq} \quad (2)$$

The total objective function of a population is equivalent to the sum of the OFS for all the strings in the population which is given by:

$$f_{tot} = \sum_{i=1}^{P} f_i = \sum_{i=1}^{P} \sum_{k=1}^{M} \sum_{p=1}^{m_k} \sum_{q=1}^{m_k} V_{ikp} V_{ikq} D_{ikpq} \quad (3)$$

The probability of selecting a string is the ratio of the objective function of that particular string over the total objective function of the population which is given as

$$p_i = \frac{f_i}{f_{tot}} \quad (4)$$

If we chose Eq. (4) as our fitness function we may fail to select strings that have good substring matches because some strings have one or more substrings which have a very high OFSS for some models although these strings may have a very low OFSs. Therefore, we have chosen a fitness function that selects strings with high OFS as well as strings with low OFSs that have a few substrings with high OFSSs.

The objective function of a substring, for a particular model, in a population is equivalent to the sum of the OFSS for all the substrings belonging to that model in the population which is given by:

$$f_k = \sum_{i=1}^{P} f_{ik} = \sum_{i=1}^{P} \sum_{p=1}^{m_k} \sum_{q=1}^{m_k} V_{ikp} V_{ikq} D_{ikpq} \quad (5)$$

The probability of selecting a substring representing a particular model is

$$p_{ik} = \frac{f_{ik}}{f_k} \quad (6)$$

The fitness function F_i for a string used in our algorithm is the sum of the probability of the string and the probabilities of all the substrings in that particular string which is given by:

$$F_i = p_i + \sum_{k=1}^{M} p_{ik} \quad (7)$$

3.4 Genetic operators

Our genetic algorithm is composed of four operations : reproduction, crossover, mutation, and rotation. Three genetic operators, crossover, mutation, and rotation are developed in order to find the globally optimal matching points for the model-based object recognition task. In the reproduction process, individual strings are selected according to the value of their fitness function F_i. After selection of strings, substrings belonging to the same model are recombined using genetic operators in order to generate new strings. Strings with high fitness values have higher probabilities of selection; therefore, these strings produce more offsprings in the reproduction process than the strings with lower fitness values. A simple biased roulette wheel is used to select the strings in our simulations.

3.4.1 Crossover

A pair of mated strings, called parent strings S_i^P and S_j^P, produce two tentative strings called offsprings S_i^O and S_j^O under crossover operation. Crossover operation is only allowed to be performed between a pair of substrings. First a crossover point c_p along the substring is randomly selected, having a range between 1 and m_k, then a crossover length c_l is randomly selected within the range of 1 and $m_k - 1$. Two new substrings are created by exchanging all the values between the positions c_p

and $(c_p + c_l - 1) \bmod m_k$ of the substring S_{ik}^P in the parent string S_i^P and the substring S_{jk}^P in the parent string S_j^P. This crossover operation is applied to all the substrings of the parent strings. We investigated two forms of heuristic crossover operators such as a block crossover and a gene crossover. Let B_{ik}^P represent a block of the substring S_{ik}^P of length c_l and $B_{ik}^P(l)$ represent the *lth* value of the selected block. The block crossover operation will create the substring S_{ik}^O by replacing the block B_{ik}^P of S_{ik}^P with the block B_{jk}^P of S_{jk}^P only if the contents of the resulting substring S_{ik}^O are still ordered after the replacement. The reason for this condition is that each substring is structured like a ring and the values in the substring are ordered according to the feature numbers. If the conditions for a direct block replacement is not satisfied then a gene crossover operation is done by exchanging one value at a time for all the elements in the blocks. If the replacement of $B_{2k}^P(l)$ with $B_{1k}^P(l)$ does not preserve the order of assigned values in the substring S_{2k}^O then $B_{1k}^P(l)$ is placed in the *lth* position of the block B_{2k}^P and the substring is reordered making minimal changes in S_{2k}^O. If the newly assigned value by an individual gene replacement is the same with any other values in the substring, then values which are the same as newly assigned value are discarded.

3.4.2 Mutation

Although the reproduction process and the crossover operators will search the solution space effectively, occasionally they may lose some useful solution patterns. Mutation operator will protect against such an irrecoverable loss and will avoid the algorithm to get trapped into a local minimum and will enable it to jump to the global minimum. Mutation is a process of finding a new search space by changing the value of a randomly chosen position (mutation point) within a substring which is also chosen at random. The *jth* element of the S_{ik}, $S_{ik}(j)$, can be changed to any value between $S_{ik}(j-1)$ and $S_{ik}(j+1)$ or to 'x' in order to preserve the order of the feature points assigned in a substring.

3.4.3 Rotation

There may be some strings which have a very low fitness value, in spite of good structure, because of the wrong positioning of each value even though the crossover and mutation operators have searched the solution space effectively. However, such strings may have a high fitness value if the content of the string is correctly reordered by using a rotation operator. Using a rotation operator, a solution vector $(a_1, a_2, \cdots, a_m)$ can be rotated into another solution vector $(a_i, \cdots, a_m, \cdots, a_{i-1})$ in order to get a higher fitness value.

4. Experimental Results

The proposed genetic algorithm has been tested on scenes with several overlapping objects. In this paper, three key images are used to generate the global model graph for testing the algorithm. Fig. 5 shows the prototype objects for the key images where feature points are marked as white cross. The number of feature points for the key images are 18, 13, and 11 which correspond to #1, #2 and #3 model graphs, respectively. There are a total of 42 feature points in the global model graph. Input scenes, which consists of two and three overlapping keys are shown in Fig. 6. The number of feature points in the input graphs are 22 and 30 for two and three overlapping objects, respectively. The input graph may have feature points which are falsely generated by the overlapping parts of two or more objects and may not have any matching feature points in the global model graph because of the occlusion of feature points in the input scene graph by other objects.

(a) Model image : key 1 (b) Model image : key 2 (c) Model image : key 3

(Fig. 5) Model images; (a) key 1, (b) key 2, (c) key 3

(a) Input image : two overlapping keys of key 1 and key 2 (b) Input image : three overlapping keys of key 1, 2, and 3

(Fig. 6) Input images; (a) two overlapping keys, (b) three overlapping keys

An initial population is randomly generated using random numbers, such that the assigned values to each string are in an increasing order. The fitness value for each string in the initial population is evaluated by the Eq. (7). Parents for the next generation are selected in accordance with the relative fitness values. Genetic operators such as crossover, mutation, and rotation are applied to selected strings to generate offsprings for the next generation. Generation of a new population under genetic operation continues until termination condition is satisfied. After termination, the string which has the highest fitness is selected as the best solution. In the genetic algorithm there are a number of parameters such as the number of strings in the population, number of generations(termination condition), probability of crossover, probability of mutation and the probability of rotation, the typical values used for these parameters in our experiments are 50, 40, 0.8, 0.03 and 0.03, respectively.

The final string contains the matched points between the input graph and the global model graph. Most of them are correctly matched pairs but there may still exist some mismatched points. In order to eliminate these mismatched points, a complete graph is constructed based on the feature values and the compatibility values of matched feature points. Let a graph $G(V, E)$ consists of a set of vertices V with a set of edges E of unordered pairs of the form (i, j) or (j, i) where i and j are the nodes in V. A node in our complete graph consists of a pair of matched points between a model graph and the input graph. We say that an edge between the nodes i and j is connected, if the compatibility measures(relational properties) between two nodes are less than or equal to a predefined threshold. All relational properties are tested when edges are connected. A graph is said to be complete if there exists an edge (i, j) for every pair of vertices i and j. If a graph is complete, all matched points are correctly matched and if a graph is not complete, then some matched points are mismatched points. When a graph is not complete, we remove a node which has the smallest number of connected nodes and a new graph is constructed with the remaining nodes. These steps are continued until a complete graph is obtained such that the remaining nodes of the complete graph are perfectly matched. Outputs of the matching process are correct matching points between model graph and input graph.

For the purpose of comparison with Hopfield neural network approach [1], which is an optimization method in object recognition, we conducted the same experiments using the same model and input test images. The matching process using Hopfield network is characterized as minimizing an energy function given by

$$E = -\frac{1}{2}\sum_{i=1}^{M}\sum_{k=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{N} C_{ikjl} V_{ik} V_{jl} + \sum_{i=1}^{M}(1-\sum_{k=1}^{N} V_{ik})^2 + \sum_{k=1}^{N}(1-\sum_{i=1}^{M} V_{ik})^2 \quad (8)$$

where N is the number of nodes in the global model graph and M indicates the number of nodes

in the scene graph. V_{ik} is analogous to a binary state variable of a neuron, and takes value 1 when the *ith* feature point in the input image matches the *kth* feature point in the model. The interconnection weights between the neurons are represented in terms of the compatability measure C_{ikjl} between the feature points.

Ten experiments were conducted in order to demonstrate our proposed method with different seed numbers. The matching rate, which is the ratio of the number of correctly matched points over the number of feature points in the input scene, is used. The average matching rates of genetic algorithm technique are 95% and 93% for two and three overlapping key objects, respectively. On the other hand the corresponding average matching rates for the Hopfield network are 52% and 51% for two and three overlapping objects, respectively. The matching rates of neural networks are worse than those of genetic algorithms. This is due to the Hopfield network settling into a locally stable state. Examples of detailed matching results with 2 overlapping keys and 3 overlapping keys are shown in Table 1 and 2, respectively, where the numbers in the first row are the input feature points and each entry, $x(y)$, in the second and third rows represents the matching feature point y in the model x. The symbol '*' represents missing points and '-' is the hidden or extra points due to noise or occlusion.

After finding matched points, he transformation from the model to the object is calculated using matched points. Then the model is mapped by the appropriate coordinate transformation onto the input scene. By superimposing each model on top of the scene and comparing the overlapping area of object with the area of model we can recognize each object. Overall recognition rate of genetic algorithm is 100% and that of neural network is 86%.

5. Conclusions

A two-dimensional object recognition technique is proposed for partially overlapping objects. Object recognition is formulated as a graph matching problem. This graph matching problem for model-based object recognition is then solved by a genetic algorithm. The proposed matching technique uses the implicit parallelism of genetic algorithms to globally match

〈Table 1〉 An example of matched feature points of 2 overlapping keys by the genetic algorithm (GA) and the Hopfield neural network (NN) [1]

input	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GA	-	-	2(1)	-	1(1)	1(2)	1(3)	1(4)	1(5)	1(6)	1(7)	1(8)	1(9)	1(10)	1(11)
NN	-	-	*	-	*	1(2)	1(3)	1(4)	1(5)	*	1(7)	1(8)	*	*	1(11)

input	16	17	18	19	20	21	22
GA	-	-	2(8)	2(9)	2(10)	2(11)	*
NN	-	-	*	*	*	2(11)	2(12)

〈Table 2〉 An example of matched feature points of 3 overlapping keys by the genetic algorithm and Hopfield neural network [1]

input	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GA	1(9)	1(10)	1(11)	-	1(13)	1(7)	-	3(9)	3(10)	3(0)	3(1)	3(2)	3(3)	-	-
NN	1(9)	1(10)	*	-	*	*	-	*	3(10)	3(0)	*	3(2)	3(3)	-	-

input	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
GA	-	-	-	-	-	-	-	-	-	2(11)	2(12)	2(0)	-	-	1(8)
NN	-	-	-	-	-	-	-	-	-	2(11)	2(12)	*	-	-	*

all the objects in the input scene against all the model scenes. A representational structure for the strings is proposed and some genetic operators such as crossover, mutation, and rotation are developed in order to generate good population during the generation process and to find nearly optimal solution. An appropriate fitness function, which represents all the constraints imposed by the matching process was formulated in order to select the best parents and thus to produce good offsprings. In our models, a unary constraint such as the angle of the corner and a binary constraint, namely the relational distance between two nodes, are used. Through simulation results we show that the proposed matching algorithm can apply model-based object recognition task. The algorithm presented here can very easily generate good results if the low level features can be extracted in a reliable way.

References

[1] N. M. Nasrabadi and W. Li, "Object recognition by Hopfield neural network," *IEEE Trans. Syst., Man and Cybern.*, Vol.21, No.6, pp.1523-1535, Nov. 1992.

[2] N. M. Nasrabadi and C. Y. Choo, "Hopfield network for stereo vision correspondence," *IEEE Trans. Neural Networks*, Vol.3, No.1, pp.1-9, Jan. 1992.

[3] A. Rosenfeld, R. A. Hummel and S. W. Zucker, "Scene Labeling by relaxation operations," *IEEE Trans. Syst, Man and Cybern.*, Vol.6, No.6, pp.420-433, June 1976.

[4] S. T. Barnard and W. B. Thompson, "Disparity analysis of images," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol.PAMI-2, pp.333-340, July 1980.

[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[6] M. W. Koch and R. L. Kashyap, "Using polygon to recognize and locate partially occluded objects," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol.PAMI-9, pp.483-494, July 1987.

[7] N. Ayache and O. D. Faugerus, "HYPER : A new approach for the recognition and position of two-dimensional objects," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol.PAMI-8, pp.44-54, Jan. 1986.

[8] D. H. Ballard, "Generalizing the Hough transform to detect arbutrary shapes," *Pattern Recognition*, Vol.13, pp.111-122, 1981.

[9] W. A. Perkins, "A model-based vision system for industrial parts," *IEEE Trans. Comput.*, Vol. C-27, pp.126-143, Feb. 1978.

[10] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man and Cybern.*, Vol.SMC-9, No.1, pp.62-55, Jan. 1979.

[11] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," CGIP, Vol.1, No.3, pp.244-256, Nov. 1972.

이 진 호

e-mail : jhlee@bear.kyungil.ac.kr

1974년 영남대학교 전자공학과 졸업(공학사)

1981년 영남대학교 대학원 전자공학과 계산기 전공(공학석사)

1996년 영남대학교 대학원 전자공학과 전산공학 전공(공학박사)

1979년~현재 경일대학교 공과대학 컴퓨터공학과 교수

관심분야 : 데이터 압축,프로그래밍 언어,객체지향 시스템

박 상 호

e-mail : spark@anu.andong.ac.kr

1979년 경북대학교 전자공학과(공학사)

1981년 영남대학교 대학원 전자공학과(공학석사)

1989년 Syracuse University(M.S.)

1995년 State University of New York at Buffalo(Ph.D.)

1996년~현재 안동대학교 정보통신공학과 조교수

관심분야 : 멀티미디어통신, 이동통신