Journal of Information Processing Vol. 18 164–174 (Apr. 2010)

Regular Paper

Evaluations of Feature Extraction Programs Synthesized by Redundancy-removed Linear Genetic Programming: A Case Study on the Lawn Weed Detection Problem

Ukrit Watchareeruetai,[†1] Yoshinori Takeuchi,[†1]
Tetsuya Matsumoto,[†1] Hiroaki Kudo[†1]
and Noboru Ohnishi[†1]

This paper presents an evolutionary synthesis of feature extraction programs for object recognition. The evolutionary synthesis method employed is based on linear genetic programming which is combined with redundancy-removed recombination. The evolutionary synthesis can automatically construct feature extraction programs for a given object recognition problem, without any domain-specific knowledge. Experiments were done on a lawn weed detection problem with both a low-level performance measure, i.e., segmentation accuracy, and an application-level performance measure, i.e., simulated weed control performance. Compared with four human-designed lawn weed detection methods, the results show that the performance of synthesized feature extraction programs is significantly better than three human-designed methods when evaluated with the low-level measure, and is better than two human-designed methods according to the application-level measure.

1. Introduction

Designing a feature extraction program for a given object recognition problem is a difficult, time-consuming task. Usually, it is performed by human experts who have to consider what features are appropriate for the problem at hand and how to extract such features from images. Human experts generally carry out this task based on their knowledge and experience, and sometimes under time constraints. This implies that the entire feature space cannot be explored, and it is possible that unconventional but potential features may be ignored.

Up to now, many researchers have attempted to cope with the difficulties in designing not only feature extraction programs but also various processes, including performing object recognition itself. Most methods adopt evolutionary algorithms (EA)[1)] — a set of powerful optimization/search strategies inspired by the biological evolution found in nature — to automatically synthesize programs. Many approaches have been proposed with various EA techniques, such as genetic algorithms (GA)[2)], tree-based genetic programming (GP)[3)–6)], graph-based GP[7),8)], and linear GP[9)].

We focus on an approach based on linear GP[10),11)] because of the following advantages: 1) Its representation is simple but powerful enough to represent graph-structure-based programs, which are more general than tree-based programs. 2) There is an algorithm for removing (structural) introns — the instructions that have no effect on the program output — at run-time[10)]. Therefore we can avoid wasteful execution of such instructions and greatly accelerate the evolutionary process. 3) There is a canonical transformation that converts redundant representations into a canonical form in which redundancies are removed[12)]. This enables us to improve the performance of the evolution process by various techniques[13)].

Here, we present a way to improve the search performance of linear GP by using redundancy-removed (RR) recombination[⋆1]. The key idea of this RR recombination is to avoid searches for already discovered programs . In this paper, we will show a result that indicates an advantage of the linear GP with RR recombination over linear GPs with conventional recombinations. Moreover, we mainly focus on the following important question: Are programs synthesized by the linear GP better than, comparable to, or worse than human-designed programs? The main objective of this work is to evaluate the programs synthesized by linear GP based system with RR recombination, and compare them with conventional (human-designed) methods described by related works. We have conducted an experiment with a lawn weed detection problem[15),16)] — to detect weed areas from a lawn background to perform weed control automatically. Evaluation was performed using both a low-level measure, i.e., segmentation accuracy, and an application-level measure, i.e., weed control performance.

†1 Department of Media Science, Graduate School of Information Science, Nagoya University

⋆1 Redundancy-removed recombination was first presented in Ref. 14).

Experimental results show that the synthesized program is significantly better than three human-designed lawn weed detection methods when evaluated with the low-level measure and is better than two human-designed methods when the application-level performance measure is applied.

The rest of this paper is organized as follows. Section 2 describes the linear GP based synthesis of feature extraction programs. Section 3 explains the redundancies in linear GP, canonical transformation and RR recombination. Section 4 describes the target problem and human-designed methods that are compared against. Section 5 shows our experimental results. Section 6 concludes the paper.

2. Linear GP Based Synthesis of Feature Extraction Programs

The overview of the evolutionary system for synthesizing feature extraction programs is shown in **Fig. 1**. At the beginning of the evolutionary process, a population of feature extraction programs is randomly generated and is used as an initial population. Then all programs in the population are evaluated based on a set of training images and ground truths (provided by a user). In the evaluation, leave-one-out cross-validation [17] is adopted, and the average segmentation (or classification) accuracy is used as the performance measure (or fitness value) of programs. Based on the evaluation, programs that have better performance are more likely to survive and produce offspring. The produced offspring population is considered to be the current population and is evaluated again. This evolution process continues from generation to generation until termination criteria are satisfied. Afterward, the program that gives the best performance in the current population is considered to be the output of the feature extraction program synthesis system.

We adopt a linear GP representation [10),11)] in this work. In particular, a feature extraction program is represented by using a fixed-length sequence of basic image

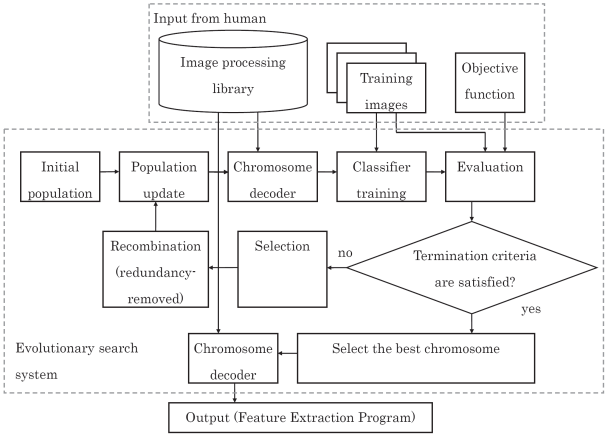


Fig. 1 Overview of evolutionary synthesis of feature extraction programs.

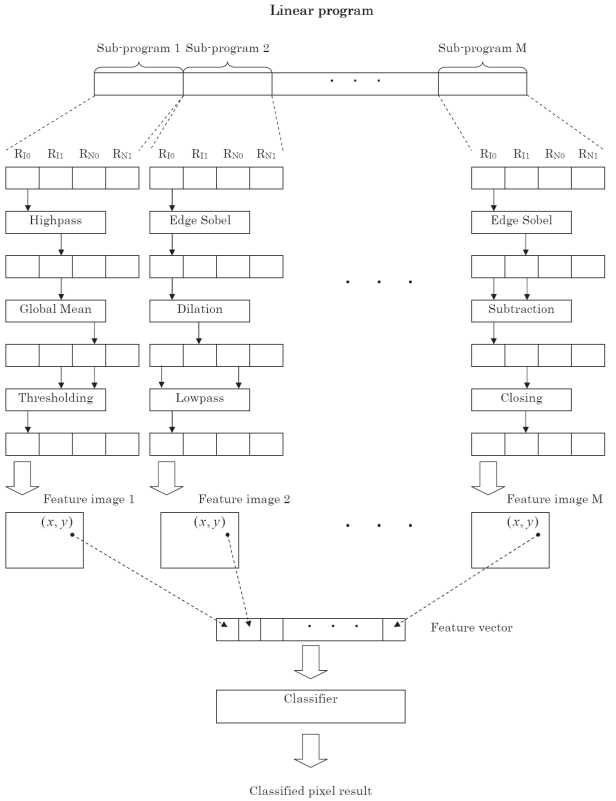


Fig. 2 Linear GP representation with sub-program structure.

processing operations, e.g., thresholding, filtering, edge detection, and histogram equalization (see Appendix). These basic image processing operations are used as primitive operations (POs). In the execution process, each instruction is sequentially decoded and executed based on a set of shared registers, analogous to the program execution on modern microprocessors. Namely, an instruction fetches inputs from registers, processes them, and stores the processed result into a register. Two types of registers are used: image registers (R_I) and numerical registers (R_N). Each instruction is encoded by using four components, i.e., one operation code (op-code) that describes the PO to be executed and three arguments that describe either the indexes of the input and output registers or a parameter of the instruction. In this work, one linear program consists of multiple sub-programs (**Fig. 2**). Each sub-program is executed independently of each other and generates one feature image. For each pixel, the intensity values in all feature images are integrated to construct a feature vector. A classifier is adopted to decide which class that pixel belongs to, based upon the feature vector. Here, we adopt a Bayesian classifier with a histogram approximation method [17], in which pattern distributions are approximated by using histograms.

3. Linear GP with Redundancy-removed Recombination

3.1 Redundancies in Linear GP and Canonical Transform

Here we will describe the causes of redundancies in linear GP representation: 1) One of the causes is the existence of introns — the instructions that do not affect the program output [⋆1]. While introns change the linear GP representations, they do not change the program output. 2) A protection mechanism that prevents the execution of undefined operations or references to undefined registers is needed to eliminate redundancies. For example, if we use 10 POs to construct programs, we need at least four bits to represent all POs. However, since four bits can represent more than 10 numbers, undefined operations may result. In this case, a simple mechanism that replaces the original op-code by the op-code modulo N_{PO} (N_{PO} is the number of POs) can be adopted to prevent the execution of undefined operations. This causes redundant representations because the op-codes of 0 and 10 (or 1 and 11, and so on) map to the same operation. 3) Different instruction orders also cause redundancies if there are two (or more) instructions running in parallel which do not depend on each other. For example, **Fig. 3** (left) shows a case where the first two instructions can be reordered without any effect on the program output. 4) Generally, an identical program can be constructed that uses registers in alternative ways. Figure 3 (right) shows an example of two instruction sequences that represent two identical programs that use registers in different ways. 5) The use of sub-program structure, as described in Section 2, also causes redundancies because a change in the order of image features does not affect the classification result, even though the representations are different.

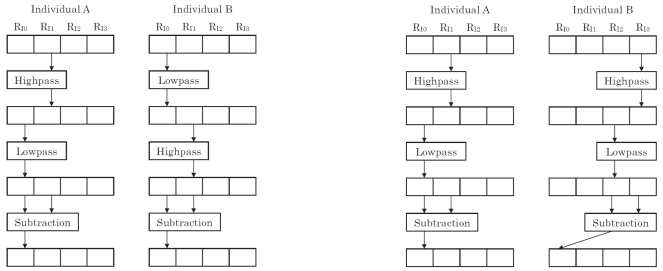


Fig. 3 Examples of redundancies: two instruction sequences containing the same operations but in different order (left) and two instruction sequences assigning register usage in different ways (right) (©2008 IEEE).

Recently, we have proposed a transformation that converts linear GP representations into canonical forms in which such redundancies are removed [12]. It can be briefly described as follows (more details can found in Ref. 12)).

(1) Intron removal: we use an algorithm described in Ref. 10) to remove structural introns from the instruction sequence under consideration. The algorithm searches backwards from the last instruction through the dependent instructions, and marks them as *effective* instructions because they relate

⋆1 In linear GPs, introns can be divided into two types: *structural* and *semantic* introns [10]. In this work, we consider only structural introns because semantic introns highly depend on the problem to be solved, and are very difficult to be detected. Although a semantic intron removal method has been described in Ref. 10), the method is not practical for our problem due to a huge number of iterations and the method does not guarantee the removal of all semantic introns. However, if there is an efficient algorithm to remove semantic introns, it can be added into the *intron removal* step of the canonical transformation. In the remaining sections, the term *intron* refers to structural introns only.

to the data flow of the program output. The remaining unmarked instructions are considered to be structural introns and are removed from the sequence.

(2) Conversion to modulo form: we replace the original op-code and arguments by their modulo forms (the op-code modulo N_{PO} and argument modulo N_{reg}, where N_{reg} is the number of registers of the corresponding data type).

(3) Instruction reordering: we search backwards from the last *effective* instruction through the dependent instructions based on a depth-first-search scheme. If we cannot search further from an instruction, we will assign it the current order and go back to the previous instruction, then repeat this step (the current order is increased by one). By doing this, different individuals representing the identical program will be reordered in the same way.

(4) Register reassignment: starting from the first *effective* instruction, we identify that the content stored in registers and until which instruction cycle they are needed. Then we set a write-protected flag on each corresponding register until the execution cycle of that instruction. Next, we assign the input register index based on the data flow but assign the non-write-protected register that has the lowest index as the output register. We then go to the next instruction and repeat this step.

(5) Sub-program reordering: we compare the *effective* lengths of sub-programs, and reorder them based on ascending order in length.

Once we know the canonical forms of two individuals, we can easily verify whether or not they represent identical programs. We can achieve this by comparing the lengths of the two canonical forms. If they have equal lengths, we then compare their (genotypic) contents byte-by-byte. The two individuals that have exactly the same canonical form will represent identical programs.

3.2 Redundancy-removed Recombination

The redundancy-removed (RR) recombination [14] exploits the canonical form to verify whether a generated offspring represents a previously discovered program earlier in the evolutionary search. In particular, we store all canonical forms of the programs that have already been discovered in memory. Once an offspring is generated, it will be transformed into its canonical form, and compared with the other canonical forms in memory. If it matches a canonical form in memory, it means that the generated offspring represents a program that was previously discovered. If this happens, the program will not be allowed to survive to the next generation. We call such an offspring a redundant offspring. The RR recombination consists of two operators: RR crossover and RR mutation.

For RR crossover, a conventional crossover such as one-point, two-point or uniform crossovers is applied to a pair of parents to generate offspring (here we consider only one offspring per crossover). If the generated offspring is redundant, it will be ignored, and parents will be (randomly) re-selected to generate new offspring instead. If the new offspring is still redundant, this process will be repeated. In the case that the maximum number of repeats (20 in our work) is reached and no non-redundant offspring is generated, the latest offspring is allowed to survive in the next generation. For RR mutation, we apply a mutation operation on the same offspring repeatedly until it becomes a non-redundant offspring.

3.3 Comparison with Conventional Recombinations

Here, linear GPs with different recombinations, i.e., two-point crossover, parameterized uniform crossover [18] (with gene exchanging probability of 0.2), and RR-recombination, were evaluated. We used an artificial problem as the test problem. In particular, we defined an image processing program, based on the POs provided, that transforms an input image into an output image. The goal is to search for the defined image processing program in the search space. In this case, there is at least one correct solution that exists in the search space. Consequently, we can observe whether the linear GPs can synthesize the correct solution. The defined image processing program consists of three POs, i.e., the Sobel operator, adaptive mean thresholding, and morphological opening with a square structural element of size 3×3, connected in sequential order. The parameters of the linear GPs are shown in **Table 1** ⋆1. Each approach has been executed for 20 trials with different seeds (using five training images of size 160

⋆1 For linear GP with the RR recombination, we used a large tournament size (10) to balance the exploration and exploitation powers, which is a key issues in the success of EAs[1]. However, the use of a large tournament size for the two conventional recombinations causes more redundancies, resulting in poorer performance.

Table 1 Parameter settings.

Parameter	Setting	Parameter	Setting
Population size	100	Number of image registers	6
Maximum generation	500	Number of numerical registers	6
Crossover rate	0.74	Mutation rate	0.25
Reproduction rate (elitist)	0.01	Tournament size	
Length of sub-program	10	- conventional crossover	2
Number of POs	51	- RR recombination	10

Table 2 Comparison of two-point crossover, parameterized uniform crossover, and redundancy-removed recombination.

Method	Two-point	Parameterized uniform	Redundancy-removed
Number of hits	4	5	11
Average best fitness	95.74	96.01	97.79
Average generations	436.60	429.10	277.45
Average generations (only hit trials)	183.00	216.40	95.36

$\times$ 120 pixels).

The experimental results are shown in **Table 2**. We count the number of trials required until the linear GPs can find the correct solution (number of hits), and also report on the average fitness and average generations needed. The results show that the linear GPs with the two-point and parameterized-uniform crossovers attain the correct solution for only four and five trials, respectively, while the linear GP with the RR recombination achieves the correct solution for 11 trials. Moreover, the linear GP with the RR recombination can converge to the correct solution within a smaller number of generations. This shows an advantage of the RR recombination over the conventional recombination methods.

4. Lawn Weed Detection Problem

4.1 Dataset

Image processing based weed detection methods have been widely studied, especially for agriculture fields. Automatic weed control systems exploit a weed detection method to locate the area of weeds in a field so that the systems can accurately remove the weeds from the field, e.g. by spraying herbicide only onto the area of detected weed instead of spraying in the entire area. We are interested in weed detection in lawn fields. The lawn weed database used in this experiment is the database 1 used in Ref. 15). It consists of 30 images (note that among them, five images are lawn background images that contain no weeds). The image size is 640×480 pixels, covering lawn area of size 274×205 mm. **Figure 4** shows an example of a lawn image that includes some weeds.

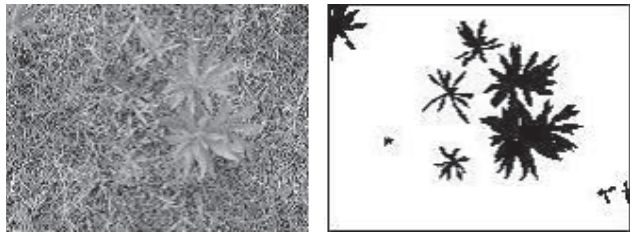


Fig. 4 An example of a lawn weed image and its ground truth.

4.2 Compared Methods

Until now, various methods have been proposed for weed detection in lawns. Ahmad, et al.[19] proposed a weed detection method based on gray-scale uniformity analysis (denoted by UA) which distinguishes weed surfaces from lawn surfaces based on the difference in gray-value distributions of both surfaces. Then, as a post-processing process, a blob inspection method is exploited to remove misclassified blobs. Previously, we have proposed two lawn weed detection methods [15]. One is a Bayesian classifier based method (denoted by BC) which detects weeds using two textural features, i.e., mean and variance of edge strength. The other one is a morphological operation based method (denoted by MO) which segments weeds from a lawn background by using morphological image processing techniques such as closing and opening. Moreover, in Ref. 16), the BC method is slightly modified. In particular, a support vector machine is adopted instead of a Bayesian classifier. We found that it is better than the BC method in some situations. We denote this modified method by SVM. In this work, all four human-designed methods, i.e., UA, BC, MO, and SVM methods, are compared with the feature extraction programs synthesized by the linear GP with RR recombination.

5. Experimental Results

Here we evaluate and compare the performance of synthesized programs with the lawn weed detection methods described in the previous section. We adopt two kinds of performance measures. One is a low-level measure — segmentation accuracy (Section 5.1). The other is an application-level measure which measures the performance of weed control (Section 5.2).

5.1 Low-level Measure: Segmentation Accuracy

We run the linear GP with the RR recombination with the parameters described in Table 1 to synthesize feature extraction programs for the lawn weed detection problem. Segmentation accuracy, which is the ratio of the number of correctly segmented pixels to the total number of pixels, was used as the fitness function for the evolutionary process. We chose the number of features (sub-programs) to be two so that it is equal to the number of features used by the other methods under consideration (BC and SVM). Ten independent GP runs were conducted with different seeds (using five training images). Once we obtained the best program from each trial, we compared them with the four human-designed methods. To perform a statistical test, we divided the database of 30 images into six folds of five images, and performed the experiments six times so that a different fold is the training set each time and the remaining folds are used for testing.

Table 3 Segmentation accuracies (%) of the human-designed methods and programs synthesized by 10 GP trials based on test sets (over six experiments).

Test set #	1	2	3	4	5	6	AVG	STD
MO	98.60	98.40	98.49	98.50	98.46	98.33	98.46	0.43
UA	98.45	98.26	98.27	98.35	98.26	98.14	98.29	0.10
BC	98.41	98.11	98.21	98.25	98.23	98.07	98.21	0.12
SVM	97.62	97.19	97.32	97.64	97.67	96.91	97.39	0.31
Trial #1	98.50	98.26	98.32	98.41	98.28	98.06	98.30	0.15
Trial #2	98.64	97.57	98.47	98.23	97.90	97.72	98.09	0.43
Trial #3	98.25	96.88	98.30	98.10	96.96	93.49	97.00	1.84
Trial #4	98.51	98.02	98.01	98.40	97.91	97.98	98.14	0.25
Trial #5	**98.75**	98.39	98.48	97.89	**98.52**	**98.44**	98.41	0.28
Trial #6	97.83	98.34	98.25	98.51	98.42	98.27	98.27	0.23
Trial #7	98.37	97.70	98.40	97.86	98.26	97.82	98.07	0.31
Trial #8	98.66	**98.46**	**98.51**	**98.58**	98.47	98.23	**98.48**	0.15
Trial #9	98.34	98.28	98.31	98.09	98.31	98.11	98.24	0.11
Trial #10	97.94	97.90	97.79	98.12	97.81	97.60	97.86	0.17

Table 3 shows the segmentation accuracy of the synthesized programs and the human-designed methods. Note that the parameters (such as window size, threshold values, and structure element size) of the human-designed methods were adjusted and only the best results were shown. The synthesized programs can correctly segment the weed areas from a lawn background with average accuracies of 97–98% for the test sets. Among 10 trials, the best one, with an average segmentation accuracy of 98.48%, is from GP trial #8 (**Fig. 5**, top left), followed by trial #5 (Fig. 5, top right) and trial #1 (Fig. 5, bottom). The average segmentation accuracy of trial #8 is higher than that of the four human-designed methods. We conducted a statistical t-test [20] to check whether the average segmentation accuracy obtained by GP trial #8 is significantly higher (the alternative hypothesis H_1). The p-values, i.e., probabilities that the null hypothesis H_0 (no difference between the two average values) is valid, are shown in **Table 4**.

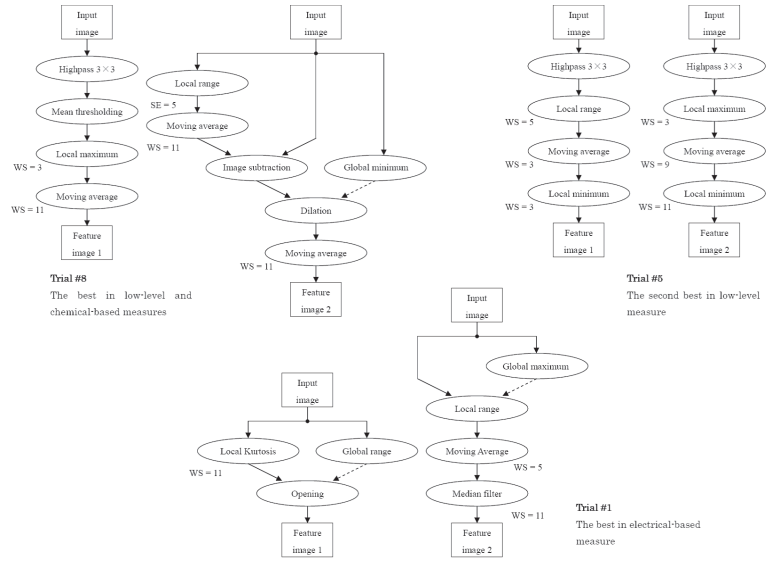


Fig. 5 Flowcharts of the synthesized feature extraction programs of trials # 1, 5, and 8 (WS stands for window size, the solid-lines represent image data flow, and the dashed-lines represent numeric data flow).

Table 4 Statistical t-test (p-value) between the human-designed methods and the GP trial #8.

MO	UA	BC	SVM
0.24	1.65×10^{-4}	1.19×10^{-4}	2.18×10^{-5}

Table 5 Ranking based on the segmentation accuracies (the four human-designed methods and 10 GP trials).

Test set #	1	2	3	4	5	6	AVG
MO	4	2	2	3	3	2	2.67
UA	7	6	9	6	7	5	6.67
BC	8	8	11	7	9	7	8.33
SVM	14	13	14	14	13	13	13.50
Trial #8	2	1	1	1	2	4	1.83
Trial #5	1	3	3	12	1	1	3.50
Trial #1	6	7	6	4	6	8	6.17

The results show that GP trial #8 is significantly better than the UA, BC, and SVM methods.

We then ranked for the four other methods under comparison and the 10 GP trials (**Table 5**). The results show that trial #8 has the best average rank (1.83), followed by the MO method, trial #5, then trial #1. The UA and BC methods have moderate ranks: 6.67 and 8.33, respectively. Also, all synthesized programs are better than the SVM method (averaged rank of 13.50) for almost all experiments. This demonstrates the success of the synthesized program over the other methods under comparison using the low-level measure.

5.2 Application-level Measure: Weed Control Performance

Here we compare the synthesized program with the human-designed methods based on their performance in simulated weed control systems. Two types of simulated weed control systems are considered. One is a chemical-based system which destroys weeds by spraying herbicide, and the other is an electrical-based system which destroys weeds by applying high-voltage spark discharges onto the weeds (see Appendix). These two systems have been described and adopted in Ref. 15). The performance of weed control in the chemical-based and electrical-based systems are shown in **Table 6** and **Table 7**, respectively. Again, the parameters of the human-designed method were adjusted for each system and the best values are shown. For the synthesized programs, we considered only the program with the best performance for each weed control situation, i.e., trial #8, which is also the best according to the low-level measure (Fig. 5, top left) for the chemical-based system, and trial #1 (Fig. 5, bottom) for the electrical-based system [⋆1].

Table 6 Average performance comparison of lawn weed detection methods for the chemical-based weed control system.

Rank	Method	Six validation sets: the average number of weeds = 66.67, the average number of weed blocks = 1615.83			
		KWR (N_{KW})	$CSPR$ (N_{CSB})	$FSPR$ (N_{FSB})	$HBRR$
1	BC	87.80% (58.50)	93.36% (1330.17)	6.64% (94.67)	91.09%
2	UA	85.00% (56.67)	95.49% (1375.00)	4.51% (65.00)	91.00%
3	Synthesis (trial #8)	83.25% (55.50)	93.80% (1396.83)	6.20% (92.33)	90.69%
4	SVM	79.07% (52.67)	98.81% (1187.00)	1.19% (14.33)	92.49%
5	MO	78.45% (52.50)	91.86% (1439.17)	8.14% (127.50)	90.21%

Table 7 Average performance comparison of lawn weed detection methods for the electrical-based weed control system.

Rank	Method	Six validation sets: the average number of weeds = 66.67		
		KWR (N_{KW})	$CSPKR$ (N_{CSPK})	$FSPKR$ (N_{FSPK})
1	SVM	75.47% (50.17)	96.95% (153.50)	3.05% (4.83)
2	BC	73.47% (48.83)	95.92% (113.50)	4.08% (4.10)
3	Synthesis (trial #1)	67.35% (44.83)	94.79% (78.83)	5.21% (4.33)
4	UA	63.88% (42.50)	95.15% (81.67)	4.85% (4.17)
5	MO	52.64% (35.00)	91.78% (55.83)	8.22% (5.00)

The main values we compared are the killed weed rate (KWR), which indicates weed destruction performance, and the number of false sprayed blocks N_{FSB} (or the number of false sparks N_{FSPK}) which describes destruction error. Ranking was done based on these values. The ranking criterion we used is the same as in our previous work [16], i.e., the method that gives errors lower than an acceptable

⋆1 Note that best individual using the low-level measure does not necessarily have the best performance for the application-level experiments because the goals of these two measures are different. For example, for the case of the electrical-based system, we do not need to obtain completed segmentation results to apply spark discharges to all weeds but we need to detect only at least one part of each weed. Better results in segmentation accuracy do not guarantee that more weeds must be detected.

value and gives higher weed destruction performance will have a better rank. The acceptable values of the chemical-based and electrical-based systems are 160 false sprayed blocks and five false sparks, respectively.

In the case of the chemical-based system, the synthesized program is ranked third among all five methods. The weed destruction performance (KWR) of trial #8 is lower than that of the best one (BC) by 4.55 points, but results in slightly lower spraying errors (false spray rate, $FSPR$). However, trial #8 is better than the MO method (which had the best low-level measure among the four human-designed methods) for both KWR and $FSPR$. Note that all methods can successfully reduce the amount of herbicide usage, i.e., they give herbicide reduction rates $HBRRs$ of higher than 90%, and these numbers are not much different from each other.

In the case of the electrical-based system, the synthesized program (trial #1) is ranked third again. Its weed destruction performance is lower than that of the SVM and BC methods by around 8 and 6 points, respectively. However it is slightly better than that of UA and is clearly better than that of MO (around 15 points). All of the methods are not noticeably different in terms of sparking errors (N_{FSPK}). Although the results suggest that the feature extraction programs, which was automatically generated without any domain-specific knowledge, could not beat all of the human-designed methods under consideration, their performances are still competitive.

5.3 Discussion

The experimental results show that the programs synthesized by the linear GP with RR recombination can outperform conventional weed detection methods when evaluated using the low-level performance measure. The statistical t-test shows that trial #8 is significantly better than three human-designed methods. The main reason for this success is that the proposed linear GP system has searched through a great number of feature extraction programs (some of them might be unconventional) based on the evolutionary search strategy. This procedure focuses on programs with better performance while still preserving global search. According to the parameter setting, 50,000 individuals are generated in one GP trial. Note that these programs are not just considered, but actually executed and evaluated so that their actual performance can be quantified. In addition, by using the RR recombination, the linear GP is forced to find new feature extraction programs that have not been discovered yet. This increases the number of different programs considered in the evolutionary search.

For the application-level evaluation, we have found that the proposed linear GP can synthesize programs that are similar in performance to, but not better than, the human-designed methods under examination. This is because the fitness function we used, i.e., segmentation accuracy, does not directly indicate the performance of weed control systems. Moreover, based on the ranking, we considered both weed destruction performance (KWR) and weed destruction error (N_{FSB} or N_{FSPK}). They are the two objectives to be optimized. This points out the need for a multi-objective optimization technique. If the feature extraction program synthesis system can optimize various objectives simultaneously, it might be possible to synthesize programs that are also better for the application-level measure.

A major concern of this system seems to be its huge computation time. However, it is possible to be executed on an inexpensive present-day desktop system. In this paper, we have run this system on an Intel Core2 Duo PC with 1,024 GB of memory. One GP trial took computation time from several dozen to around a hundred hours of CPU time (but two GP trials can simultaneously be executed). Its computation cost could be significantly reduced by using a general purpose graphics processing unit (GPGPU) for executing image processing operations [21], or by using an algorithmic reduction (e.g., as in Refs. 13), 22)).

6. Conclusion

We have described a procedure for the evolutionary synthesis of feature extraction program based on linear GP with RR recombination. It can automatically synthesize feature extraction programs for a given problem without domain-specific knowledge. A lawn weed detection problem was considered in this paper. We compared the performance of the feature extraction program synthesized by the system with the detection methods proposed in the literature. The experimental results show that the performance of the synthesized program is significantly better than three human-designed lawn weed detection methods using a low-level evaluation measure and is better than two human-designed

methods using an application-level measure. The main reason for this success is that the evolutionary synthesis system can search for a large number of different programs based on an evolutionary search strategy which considers redundancies. In the future, we plan to improve the performance of the evolutionary synthesis of feature extraction programs by using a multi-objective optimization technique so that we can simultaneously optimize multiple objectives.

Acknowledgments This work was supported by a research grant from the Hori Information Science Promotion Foundation. We thank the anonymous reviewers for their useful comments that help us improve the paper.

References

1) De Jong, K.A.: *Evolutionary Computation: A Unified Approach*, The MIT Press (2006).
2) Nagao, T. and Masunaga, S.: Automatic Construction of Image Transformation Processing Using Genetic Algorithm, *Proc. ICIP-96*, Vol.3, Lausanne, Switzerland, pp.731–734 (1996).
3) Aoki, S. and Nagao, T.: Automatic Construction of Tree-structural Image Transformations Using Genetic Programming, *Proc. ICAIP-99*, Venezia, Italy, pp.136–141 (1999).
4) Roberts, M. and Claridge, E.: A Multi-stage Approach to Cooperatively Coevolving Feature Construction and Object Recognition, *Application of Evolutionary Computing*, Rothlauf, F., et al. (Eds.), LNCS 3449, pp.396–406, Springer-Verlag (2005).
5) Trujillo, L. and Olague, G.: Automated Design of Image Operators That Detect Interest Points, *Evolutionary Computation*, Vol.16, No.4, pp.483–507, MIT Press (2008).
6) Zhang, M., Ciesielski, V. and Andreae, P.: A Domain-independent Window Approach to Multiclass Object Detection Using Genetic Programming, *EURASIP J. on Applied Signal Processing*, Vol.8, pp.841–859 (2003).
7) Shirakawa, S. and Nagao, T.: Genetic Image Network (GIN): Automatically Construction of Image Processing, *Proc. IWAIT-2007*, Bangkok, Thailand, pp.643–648 (2007).
8) Teller, A. and Veloso, M.: PADO: A New Learning Architecture for Object Recognition, *Symbolic Visual Learning*, Ikeuchi, K. and Veloso, M. (Eds.), pp.77–112, Oxford University Press (1997).
9) Krawiec, K. and Bhanu, B.: Visual Learning by Evolutionary and Coevolutionary Feature Synthesis, *IEEE Trans. Evolutionary Computation*, Vol.11. No.5, pp.635–650 (2007).
10) Brameir, M. and Banzhaf, W.: *Linear Genetic Programming*, Springer (2007).
11) Poli, R., Langdon, W.B. and McPhee, N.F.: *A Filed Guide to Genetic Programming*, Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk (2008).
12) Watchareeruetai, U., Takeuchi, Y., Matsumoto, T., Kudo, H. and Ohnishi, N.: Transformation of Redundant Representations of Linear Genetic Programming into Canonical Forms for Efficient Extraction of Image Features, *Proc. IEEE CEC-08*, Hong Kong, China, pp.1996–2003 (2008).
13) Watchareeruetai, U., Takeuchi, Y., Matsumoto, T., Kudo, H. and Ohnishi, N.: Efficient Construction of Image Feature Extraction Programs by Using Linear Genetic Programming with Fitness Retrieval and Intermediate-result Caching, *Foundation of Computational Intelligence Volume 4: Bio-inspired Data Mining*, Abraham, A., et al. (Eds.), Springer-Verlag (2009).
14) Watchareeruetai, U., Takeuchi, Y., Matsumoto, T., Kudo, H. and Ohnishi, N.: Improving Search Performance of Linear Genetic Programming Based Image Recognition Program Synthesis by Redundancy-removed Recombination, *Proc. SMCia/08*, Muroran, Japan, pp.393–398 (2008).
15) Watchareeruetai, U., Takeuchi, Y., Matsumoto, T., Kudo, H. and Ohnishi, N.: Computer Vision Based Methods for Detecting Weeds in Lawns, *Machine Vision and Application*, Vol.17, No.5, pp.287–296 (2006).
16) Watchareeruetai, U.: Lawn Weed Detection Methods Using Image Processing Techniques for Automatic Weed Control Systems, Master's thesis, Graduate School of Information Science, Nagoya University (2007).
17) Theodoridis, S. and Koutroumbas, K.: *Pattern Recognition 3rd edition*, Academic Press (2006).
18) Spears, W.M. and De Jong, K.A.: On the Virtues of Parameterized Uniform Crossover, *Proc. ICGA-91*, Belew, R.K. and Booker, L.B. (Eds.), pp.230–236, Morgan Kaufmann (1991).
19) Ahmad, U., Kondo, N., Monta, M., Arima, S. and Mohri, K.: Weed Detection in Lawn Field Based on Gray-scale Uniformity, *J. of the Japanese Society of Environmental Control in Biology*, Vol.36, No.4, pp.227–237 (1998).
20) Kanji, G.K.: *100 Statistical Tests, 3rd edition*, SAGE Publications (2006).
21) Ando, J. and Nagao, T.: Fast Tree-structural Image Processing Using GPU, *Proc. IWAIT-2007*, Bangkok, Thailand, pp.423–428 (2007).
22) Watchareeruetai, U., Matsumoto, T., Ohnishi, N., Kudo, H. and Takeuchi, Y.: Acceleration of Genetic Programming by Hierarchical Structure Learning: A Case Study on Image Recognition Program Synthesis, *IEICE Trans. Inf. Syst.*, Vol.E92-D, No.10, pp.2094–2102 (2009).

Appendix: Primitive Operations

Table 8 shows a list of basic image processing operations we used in this work.

Appendix: Evaluation of Chemical-based Weed Control Systems

Evaluation of simulated chemical-based weed control systems is described as follows.

(1) Divide the detected weed image into small blocks (block size = 30 × 16 pixels).

(2) For each block, if more than 10% of the block size area is classified as weed pixels, we call that block a weed block. All weed blocks are sprayed with herbicide.

(3) Compared with manually segmented weed area (ground truth), find the number of weeds whose area of more than 30% is sprayed (N_{KW}). These are destroyed (killed) weeds.

(4) Find the number of sprayed blocks (N_{SPB}), the number of sprayed weed blocks (N_{CSPB}), the number of sprayed non-weed blocks (N_{FSPB}).

(5) Calculate killed weed rate (KWR), correct spray rate ($CSPR$), false spray rate ($FSPR$), and herbicide reduction rate ($HBRR$) as specified by the following equations:

$$KWR = \frac{N_{KW}}{N_W}, \tag{1}$$

$$CSPR = \frac{N_{CSPB}}{N_{SPB}}, \tag{2}$$

$$FSPR = \frac{N_{FSPB}}{N_{SPB}} = 1 - CSPR, \tag{3}$$

$$HBRR = 1 - \frac{N_{SPB}}{N_B}, \tag{4}$$

where N_W is the total number of weeds in the dataset and N_B is the total number of blocks.

Table 8 A list of the primitive operations used in this work.

One-input operations	Two-input operations
image → image	**image, image → image**
highpass filter	image addition
Sobel operation	image subtraction
image negative	**image, real value → image**
mean thresholding	lowpass filter
entropy thresholding	median filter
histogram equalization	morphological dilation
image → real value	morphological erosion
global mean	morphological opening
global variance	morphological closing
global STD	local histogram equalization
global skewness	thresholding
global kurtosis	local variance
global maximum	local skewness
global minimum	local kurtosis
global median	local maximum (max filter)
global mode	local minimum (min filter)
global range	local mode
global entropy	local range
	local entropy

Appendix: Evaluation of Electrical-based Weed Control Systems

Evaluation of simulated electrical-based weed control systems is described as follows.

(1) Find a set of sparking points [16] from the detected weed image. A spark discharge will be applied to each sparking point.

(2) Calculate KWR according to Eq. (1), and correct spark rate ($CSPKR$), and false spark rate ($FSPKR$) as described by the following equations:

$$CSPKR = \frac{N_{CSPK}}{N_{SPK}}, \tag{5}$$

$$FSPKR = \frac{N_{FSPK}}{N_{SPK}}, \tag{6}$$

where N_{CSPK} is the number of sparked weed pixels (correct sparks), N_{FSPK} is the number of sparked non-weed pixels (false sparks), and N_{SPK} is the total number of sparked points.

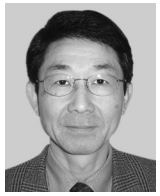
(Received June 4, 2009)
(Accepted January 8, 2010)
(Released April 7, 2010)

Ukrit Watchareeruetai received his B. Eng in Electrical Engineering from Kasetsart University (Thailand) in 2002. From 2002–2004, he worked as a research assistant at Kasetsart Signal and Image Processing Laboratory. He received the Master degree in Information Science from Nagoya University (Japan) in 2007. He is currently a doctoral candidate at the same university. His research interests include computer vision, pattern recognition and evolutionary computation. He is a student member of IEEE, ACM and IPSJ.

Yoshinori Takeuchi received his B. Eng., M. Eng. and Dr. Eng. degrees from Nagoya University in 1994, 1996 and 1999, respectively. In 1999, he was a Research Fellow of the Japan Society for the Promotion of Science. In 2000, he was a member of the Graduate School of Engineering, Nagoya University. Currently, he is an Associate Professor at the Information Security Promotion Agency, Nagoya University. His research interests include computer vision and computer audition. He is a member of IEEE, IEICE and RSJ.

Tetsuya Matsumoto received his B.E., M.E., and Dr. Eng. degrees from Nagoya University, Nagoya, Japan, in 1982, 1984 and 1996, respectively. From 1984 to 1989, he worked in Toshiba Corporation, Fuchu, Japan, where he was engaged in research and development of the control systems of nuclear power plants. In 1993, he joined the Education Center for Information Processing, Nagoya University. In 1998, he moved to the Department of Information Engineering, Graduate School of Engineering, Nagoya University. His current interests include neural networks, image processing and machine learning. He is a member of IEICE, JNNS and JSAI.

Hiroaki Kudo received his B. Eng., M. Eng. and Dr. Eng. degrees at Nagoya University, Japan in 1991, 1993 and 1996, respectively. In April 1996, he was a faculty member of the School of Engineering, Nagoya University as a Research Associate. In April 1997, he was a faculty member of the Graduate School of Engineering, Nagoya University. In April 1999, he was an Assistant Professor. In August 2000, he was an Associate Professor at the Center for Information Media Studies, Nagoya University. Since 2003, he has been a member of the Graduate School of Information Science, Nagoya University. He was a Research Fellow of the Japan Society for the Promotion of Science in 1995. His research interests include visual perception and computer vision. He is a member of IEEE, ITE, IEEJ, and IEICE.



Noboru Ohnishi received his B. Eng., M. Eng. and D. Eng. degrees from Nagoya University, Nagoya, Japan, in 1973, 1975 and 1984, respectively. From 1975 to 1986 he was with the Rehabilitation Engineering Center under the Ministry of Labor of Japan. From 1986 to 1989 he was an Assistant Professor in the Department of Electrical Engineering, Nagoya University. From 1989 to 1994, he was an Associate Professor. Since 1994, he has been a professor in Nagoya University. From 1993 to 2001, he concurrently held the position of Head of Laboratory for Bio-mimetic Sensory Systems at the Bio-mimetic Control Research Center of RIKEN. He is now in the Graduate School of Information Science. His research interests include computer vision and computer audition, robotics, bio-cybernetics, and rehabilitation engineering. Dr. Ohnishi is a member of IEEE, IEEJ, IEICE, IPSJ, SICE, JNNS, IIITE and RSJ.