2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing

A Genetic Programming Based Scheme for Combining Image Operators

Feng-Cheng Chang
Dept. of Innovative Information and Technology
Tamkang University, TAIWAN
Email: 135170@mail.tku.edu.tw

Hsiang-Cheh Huang
Department of Electrical Engineering
National University of Kaohsiung, TAIWAN
Email: hch.nuk@gmail.com

***Abstract*—Sophisticated image processing is usually nonlinear and difficult to model. In addition to the conventional image processing tools, we need some alternatives to bridge the gap between low-level and semantic level computation. This paper presents an idea of image processing scheme. We transform an image into different representations; feed the representations to the proper cellular automaton (CA) components to produce the information images; use the information images as the inputs to the combination program; and finally get the processed result. To identify the needed transforms, the CA transition rules, and the combination expression, we adopt genetic programming (GP) and cellular programming (CP) to search for the configuration. The searched configuration separates the parallelizable and sequential parts of the program. We don't enforce the linearity of the program, and it is likely that the searched result matches to the nonlinear nature of human semantics.**

***Keywords*-genetic programming; cellular programming; image processing**

I. Introduction

Computers are designed to help people doing various kinds of tasks. One of them is to find the optimal solution to a problem that does not have a closed form or even difficult to model. To find the best-fit solution based only on a set of example data, one of the popular approaches is the *evolutionary algorithms* [1]. Swarm intelligence simulates the behaviors of lives. The assumption is that we can at least construct a fitness evaluation method for a complex problem. Based on the metric, we use different strategies to search for the best solution that fits to this problem. In the past a few decades, swarm intelligence was widely used in various kinds of application domains. The typical usage is to find optimal high-dimensional solution vectors.

Due to advances in computer hardware speed and capacity, one of the techniques, *genetic programming (GP)*, becomes more and more feasible for practical cases [2][3][4]. Instead of searching for data vectors, GP searches for a *program*. That is, the structure that contains both operators and operands (values). The concept of GP was developed pretty early, but was considered as an academic tool due to the limitation of computing hardware. Recent years, some practical uses of GP emerge. Although they are sill high-complexity, they can be quite useable by adopting modified program expression (e.g., LGP) [5][6] and improved program construction (such as [7], [8] and [9]). The other evolutionary technique we are interested in is *cellular programming (CP)* [10][11][12]. It is the process to find the proper transition rules for *cellular automaton (CA)*. Researches show that CA could be used to process data in parallel with simple state-transition rules and thus suitable for hardware acceleration.

In this paper, we propose a scheme for image processing. This scheme uses GP to search for the (sub-)optimal configuration of a program. We also adopt the concept of CP to represent the parallelizable processing in order to make the found solution ready for hardware acceleration. In Sec. II, we describe the motivations why we consider image processing with GP. In Sec. III, we briefly introduce the concepts of GP and CP, which are the two major techniques in our scheme. Then, we propose the scheme and discuss the concerns in Sec. IV and Sec. V respectively. At the end, we conclude this paper in Sec. VI.

II. Motivations

Image processing techniques have been developed for a few decades. They evolve from analog image to digital image, from pixel-based operation to block-based operation, and from a single image to a multi-view image. With the growth of hardware computing capability, more and more processing targets that are closer to human semantics become feasible. Traditionally, digital image processing is treated as two-dimensional signal processing. Therefore, we may apply various kinds of tools in a signal processing problem. These tools are typically low-level and linear operators. Unfortunately, human visual system is far from linear signal processing. It involves nonlinear operations, semantic (logical) rules, and spatial/temporal correlation analysis.

Nowadays, stronger demands for semantic level image processing emerge. For example, image recognition and image retrieval are popular topics in search engines. These applications require sophisticated computation to intimate the processing inside human brain. Researches show that eyes are similar to low-level sensors. A received image is processed in different paths to produce feature signals, such as the edges. Then, very complicated process is applied to the processed signals and produces the desired semantic level information.

978-0-7695-4712-1/12 $26.00 © 2012 IEEE
DOI 10.1109/IIH-MSP.2012.58

IEEE computer society

It is not easy to figure out how human brain conducts the desired information. If we would like to intimate the process for a specific purpose, a blackbox approach is straightforward. The problem is how to find the proper configuration of the blackbox so that its behavior conforms to the given training input and output pairs. Because a configuration consists of the operations and the input values, it resembles a "program". Therefore, we are interested in using *genetic programming (GP)* to search for the configuration.

It is known that neurons are much slower than today's high-end computing units. However, human can process images quite efficiently. It is believed that parallelism in brain greatly helps the processing. Therefore, we are also interested in searching for a processing configuration that is suitable for hardware parallelism. A potential structure for parallel image processing is *cellular automaton (CA)*. It depends on a set of carefully designed state-transition rules. To design the optimal rule set, we need the *cellular programming (CP)* concepts that will be discussed in Sec. III-B.

According to the motivations, we will design a image processing scheme that incorporates GP and CP. The GP part searches for the program based on the training input/output in order to intimate the semantic processing; and the CP part searches for the parallelizable CA configuration in the program. The result is expected to be useful for both solution construction and hardware implementation.

III. GP AND CP

In this section, we briefly introduce the two major techniques integrated into the proposed scheme. The typical genetic programming (GP) and its variants are described in Sec. III-A; and the concepts and structure of cellular programming (CP) are described in Sec. III-B.

A. Genetic Programming

Genetic programming (GP) could be viewed as an specialized version of genetic algorithm (GA). An individual of GA is a bit vector representing a point in the solution space. The concept of GA is to evolve the population (a set of individuals) to search for the optimal solution. The typical process of GA is briefly described below:

1) **Initialization:** Generate the initial population by creating individuals of random bits.
2) **Selection:** Based on the given fitness function, select the survivals to form the new generation of population. If any of the individual satisfies the minimum criteria, terminate the process.
3) **Reproduction:** Apply crossover and mutation to generate the new individuals. If the termination condition is not satisfied, select and reproduce again.
4) **Termination:** Either the fitness criteria are satisfied or the maximum allowed number of generations is reached, terminate the process.

GP specializes GA in that each individual represents a "program". In other words, GP is an evolutionary computing technique that searches for a combination of operators and operands. A typical representation of a GP individual is a tree. A leaf node is the input data or constant, and a tree node is an operator. A tree-structure program is an expression, and we can easily limit the complexity of the searched program by the maximum depth of the tree. The evolution process of GP is similar to that of GA with some modifications:

- The crossover operator applies to sub-trees. The bit-level operation (without the knowledge of data representation) in GA is not applicable in GP because an operator has its specific meaning.
- The mutation operator applies to a tree node or a leaf node. When mutating a tree node, the value should correspond to a valid operator.
- It is impossible to determine the *100% fit*. In GA, we search for a solution that satisfies the fitness function. It is possible to find the exact solution given unlimited time and precision. However, we search a *program behavior* that satisfies the given training set. A training set is typically much smaller than the real data domain. Therefore, even a program produces the desired outputs for all the training data, we cannot determine whether it is over-fit or not. This issue implies that we have to use the number of generations to terminate the evolutionary process.

In addition to the tree-structured representation, some other approaches have been developed. One of the popular approach is called *Linear Genetic Programming (LGP)*. The representation of an LGP program follows the design of CPU instructions. A program is a sequence of instructions, and each instruction consists of an operation and operands (if any). The advantage of this approach is that the genetic operations apply to instruction level and usually simpler and faster. The disadvantage is that simply reuse the instructions in a linear structure is not feasible in many cases. For example, crossover instructions inside a conditional instruction cause the shift of the branching target location(s).

B. Cellular Programming

Cellular programming (CP) is the combination of *cellular automaton (CA)* and the search for the corresponding rules. CA, similar as GP, has been proposed for a few decades. The concept of CA is state-transitions based on the neighborhood:

1) Define the range of *neighbors* and the state-transition rules (may be a lookup table or a function).
2) Based on the state at time $t = k$, every cell computes its state at $t = k + 1$ in parallel.
3) The input data is processed by the given number of stages (from $t = 0..N$) and produce the final result.

Although CA considers only neighbor cells, the local data processing can achieve global effects as long as the input data go through enough evolutionary stages. Some famous applications, such as the *Game of Life*, demonstrate different usages of CA. The rule set of a specific CA application is equivalent to a program. It differs from a conventional computer program in that all the cells work in parallel with the same rule set. The sequential execution only deal with the synchronization of cell states at stage input and output. A special CA variant is called nonuniform-CA. It allows different cells to be associated with different rule sets, so that it can achieve much more sophisticated data processing than traditional CA.

While CA is a useful tool for data processing, one of the issues is how to determine the transition rules for a specific goal. Cellular programming is the concept that adopts some other optimization techniques to search for the proper rules. For example, we may adopt GA or GP to search for the rules. To determine the optimal CA rule set using CP, a few parameters should be specified:

- The number of stages to process the data: It is crucial because the number of stages affects the complexity and the accuracy.
- The criteria to terminate the training: It is almost impossible to determine whether a real "fit" program is found. We can only find the most-fit program for the training data set.

The traditional CA applications consider only a very limited number of cell states. Theoretically, it can deal with any number of states when ignoring the complexity. Some of the studies show that *continuous CA* is realizable in some applications. In this paper, we assume that continuous CA is an option for implementing the proposed scheme. Due to the limitation of time, we haven't verified the feasibility of this idea. If it is not practical, we will fall back to discrete CA to process gray-level image data.

IV. Scheme Design

Based on the aforementioned techniques, we propose the following design as an image processing structure that can be optimized by GP training process. For parallelizable processing, we use a few CP chains to produce the desired information images. The input data to the CP chains could be pre-processed images. For high-level processing, the information images are combined by the image operators that are represented as an expression tree. The CP part is used to extract the useful processing information (typically low-level), and the high-level part is a representation that fits to a certain complicated semantic context. The scheme is shown in Fig. 1.

To be intuitive, we choose the edge-enhancement problem as an example. The fundamental concept is to modify the edge pixels of the given image, and produce the visually enhanced (usually sharpened) effect. To achieve this goal, we need to determine the edges. There are many signal level tools, such as the Sobel filter and Canny operator, for

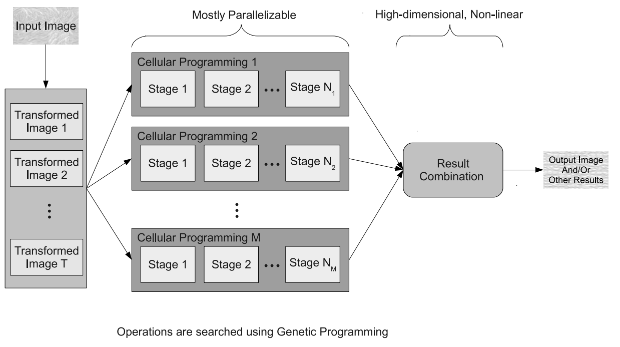

Figure 1. Image Processing Scheme with GP and CP

retrieving the potential edge image. To enhance the edges, the process depends on the given image and the adopted human visual model. The signal level tools are typically a sequence of regular image operators. On the contrary, the semantic level processing is typically a sophisticated combination of low-level tools and usually non-linear.

Suppose we solve the problem by a simple formula $I' = \alpha E + \beta I$. We need a CP-chain that takes the input image (I) and produces the edge image (E). We also need an expression tree that represents a weighted sum of the edge image and the original image for all the pixels' luminance values. Obviously, the simple approach does not match what we expect an "edge enhanced" image looks like. In fact, many image processing problems are so complicated that semantic level operations should be considered. Such kind of problems is a suitable domain for adopting GP.

When using our proposed scheme, we may prepare a few original/enhanced pairs of training images, and then engage the GP engine to search for an (sub-)optimal configuration of the CP chains and the combination operations. The result would be much different from the formula mentioned above.

V. Discussions

In this section, we discuss two practical issues in the proposed scheme. They may affect the implementation of the scheme. Until the writing of the paper, we don't have enough time to verify whether the considered solutions are effective or not. We will do simulations in the future.

A. Processing Structure Configuration

To simplify the training parameters, we assume that every CP chains contain N processing stages. Consider that some processing may require less than N stages, No-OPs may be activated only at the trail stages of a chain. This restriction enforces the normalization of the representation of a CP. We can easily prune the duplicated ones to reduce the computation complexity. There are M unique CPs for producing information images. Similarly, to simplify the computation, we represent the M-CPs as a set of CPs without empty CP.

B. Constant Parameters

A problem in this scheme is that how to determine the parameters of a given operator. We may choose the traditional approach that models a parameter as a constant input, and let the GP process to find the optimal value evolutionarily. However, image operators may require a number of parameters. Because GP is not low-complexity, it is not suitable for finding the values and operators at the same time. Our idea is to adopt conventional optimization techniques into GP. In the hybrid scheme, GP is used to determine the structure of the operators and operands. The conventional optimization techniques (such as GA and PSO) are used to determine the optimal parameters for a given expression structure.

When implementation, we need to express the combination tree as three components: the operator tree, the parameter list, and the input list. The operators tree is the signature of a given expression tree. The first time we construct a new signature, the sub-optimal parameter list is determined by the conventional optimization process. When a same tree signature occurs, we may choose to search for a better parameter list based on the given trigger probability.

VI. Conclusions

In this paper, we proposed an image processing scheme that incorporates GP and CP optimization techniques. The two techniques are used to search for a good approximation of an image processing program. The motivation is that it requires nonlinear processing for many image applications. Most of the available image processing tools are based on linear computation, and identifying the nonlinear function that matches semantics is not easy. Advances in hardware technology make evolutionary computing more practical than it was. The idea of combining GP and CP poses the following features:

- CP requires a certain optimization technique to search for the transition rules.
- GP is flexible in different optimization applications. It searches for a program (expression, function, etc) based on the training set.
- The CP chains represent the highly pararllelizable part of the program, and is the potential hardware-accelerated part for image processing.
- The result-combination component represents the non-linear part that matches human semantics.

The most important issue is the complexity of the scheme. Apparently it is not suitable for real-time processing. Its suitable applications would be those areas that require one sophisticated training and use the result repeatedly for a long time. We will address the performance issues and try to reduce the complexity in the future.

References

[1] T. Back, M. Emmerich, and O. Shir, “Evolutionary algorithms for real world applications [application notes],” *Computational Intelligence Magazine, IEEE*, vol. 3, no. 1, pp. 64 –67, february 2008.

[2] H.-S. Wong and L. Guan, “Application of evolutionary programming to adaptive regularization in image restoration,” *Evolutionary Computation, IEEE Transactions on*, vol. 4, no. 4, pp. 309 – 326, nov 2000.

[3] N. Nikolaev and H. Iba, “Regularization approach to inductive genetic programming,” *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 4, pp. 359 –375, aug 2001.

[4] N. Petrovic and V. Crnojevic, “Universal impulse noise filter based on genetic programming,” *Image Processing, IEEE Transactions on*, vol. 17, no. 7, pp. 1109 –1120, july 2008.

[5] K. Krawiec and B. Bhanu, “Visual learning by coevolutionary feature synthesis,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 3, pp. 409 –425, june 2005.

[6] ——, “Visual learning by evolutionary and coevolutionary feature synthesis,” *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 5, pp. 635 –650, oct. 2007.

[7] S. Cagnoni, F. Bergenti, M. Mordonini, and G. Adorni, “Evolving binary classifiers through parallel computation of multiple fitness cases,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 3, pp. 548 –555, june 2005.

[8] M. Zhang, X. Gao, and W. Lou, “A new crossover operator in genetic programming for object classification,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 5, pp. 1332 –1343, oct. 2007.

[9] D. Muni, N. Pal, and J. Das, “A novel approach to design classifiers using genetic programming,” *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 2, pp. 183 – 196, april 2004.

[10] J. R. Koza, *Genetic programming - on the programming of computers by means of natural selection.*, ser. Complex adaptive systems. MIT Press, 1993.

[11] W. Banzhaf, *Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications*, ser. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann Publishers, 1998. [Online]. Available: http://books.google.com.tw/books?id=1697qefFdtIC

[12] G. Adorni, F. Bergenti, and S. Cagnoni, “A cellular-programming approach to pattern classification,” in *Genetic Programming*, ser. Lecture Notes in Computer Science, W. Banzhaf, R. Poli, M. Schoenauer, and T. Fogarty, Eds. Springer Berlin / Heidelberg, 1998, vol. 1391, pp. 142–150, 10.1007/BFb0055934. [Online]. Available: http://dx.doi.org/10.1007/BFb0055934