Polymorphic Image Filters

Vojtěch Salajka*
Department of Computer Systems
Faculty of Information Technology
Brno University of Technology
Božetěchova 1/2, 612 66 Brno, Czech Republic
vojta.salajka@seznam.cz

Abstract

This paper deals with evolutionary design of a special type of image filters – polymorphic image filters. The proposed solution extends an approach for designing conventional image filters which uses Cartesian Genetic Programming. A method for accelerating the fitness function on GPU is described. We evolved polymorphic image filters capable of suppresing different combinations of noise and performing edge detection.

Categories and Subject Descriptors

I.4.3 [**Image Processing and Computer Vision**]: Filtering

Keywords

polymorphic circuits, cartesian genetic programming, image filters, evolutionary design

1. Introduction

With the emergence of polymorphic computational elements and graphene multifunctional gates [2, 6, 7], it is possible to use them as building blocks to extend some of the existing circuits, in this case image filters. Because conventional design of circuits using such elements is too complex, the approach based on evolutionary design is proposed in [1]. The purpose of this paper is to survey main concepts, solutions and results that I have obtained in my Master's thesis [1].

2. Evolutionary design of image filters

Cartesian Genetic Programming (CGP) is commonly used for evolutionary design of conventional image filters [5].

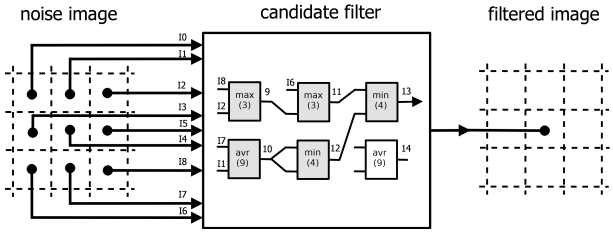

Figure 1: An example of a candidate image filter in CGP where genotype is 8,2,3, 1,7,9, 6,9,3, 10,10,4, 11,12,4, 0,11,9, 13

The evolutionary algorithm searches through the solution space for the best filter. It is an improvement over the random search. The most important concepts in evolutionary algorithms in general are the genotype – phenotype relationship and fitness function evaluation. At the beginning of the CGP algorithm a set of random genotypes is created. For each of them the phenotype is formed. It is then evaluated using a training set which contains plain or corrupted images as an input and filtered or uncorrupted images representing desired result. An illustrative example of a genotype (in this case a vector of integers) to phenotype conversion is shown in figure 1. The fitness function is designed to minimize the difference between the filtered image and the desired solution. The mean absolute error is calculated. Only the best solution is selected to generate a new generation where it is present as well. A simple point mutation operator is used to generate a new candidate filter. The process of generating and evaluating solutions is repeated for a defined number of generations.

3. Polymorphic image filters

In the presented work the CGP was used for evolutionary design of polymorphic image filters [1, 3, 4]. A polymorphic image filter contains multiple (in our case two) image filters in a single compact structure. Polymorphic computing elements (logic gates) are used to achieve this effect. These elements change their behaviour for example with temperature, voltage, external signal etc. Some real examples of such elements exist, such as polymorphic AND/OR gate, where the function depends on the temperature of environment [2]. Other examples include NAND/NOR gate developed by NASA JPL [6] and graphene logic gates [7].

The genotype was extended to make it possible to evolutionarily select a subset of a predefined set of polymor-

*Master study programme Information Technology in the field of Bioinformatics and biocomputing. Supervisor: Prof. Lukáš Sekanina. Department of Computer Systems, Faculty of Information Technology, Brno University of Technology.

Salajka, V. Polymorphic Image Filters. Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on the ACM Student Project of the Year 2012 Competition, Vol. 4, No. 4 (2012) 53-54

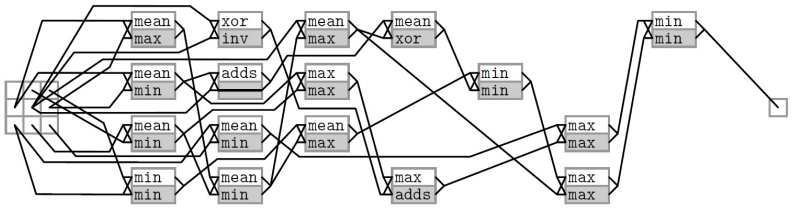

Figure 2: The best evolved filter for Gaussian noise reduction/shot noise reduction [1]

phic elements. This is useful to constraint the number of different polymorphic functions, but allows for selection of the ones best fitted for the computation at the same time. In this perspective standard logic gates are just a special case of polymorphic elements. Fitness function was extended to support two sets of images by averaging differences between achieved and desired output.

4. GPU Acceleration

The original algorithm was implemented purely in software using C++ language. An effort continued to accelerate one of the most computationally intensive parts of the algorithm – fitness evaluation – on the GPU. The OpenGL 2.1 library was used. The code of fragment shader was generated from the genotype. Then the shader was compiled and used to evaluate the candidate solution. This process had to be replicated for each candidate solution, so the acceleration proved useful only for bigger training images – around 512×512 pixels and larger in the testing environment[1]. The shader used the input image and the desired output image to compute the difference image that was read back to RAM. The summation of the differences and the rest of the fitness computation took place on the CPU.

5. Results

In order to get statistically significant results each run of CGP was repeated 20 times. In a single run 40,000 generations were produced with the population size of 5 individuals. Three polymorphic image filters were designed with the aforementioned algorithm – dilation/erosion, edge detection/shot noise reduction and Gaussian noise reduction/shot noise reduction. The achieved results in terms of the mean absolute error per pixel are in the table 1. All evolved filters were evaluated using 16 test images.

Because the filters were designed automatically by an evolutionary algorithm, their exact function in unknown. Despite of this, they generally use computational elements which we would expect. For example Gaussian noise reduction part of the third image filter uses many instances of averaging computing elements, see figure 2 (white parts of elements).

6. Conclusions

The quality of the results strongly depends on input parameters of the algorithm. These parameters are difficult to set intuitively, so it was necessary to analyze positive trends by doing initial experiments with various different settings. Generally the search space must not be both too small and not too big at the same time. If it was too small, the quality solution would not be found at all. If it was too big, the quality solution would not be found in a reasonable amount of time or the resulting image filter would be too computationally expensive itself to be useful.

[1]For the standard solution, two quad-core Intel Xeon processors were used. For the accelerated solution, Pentium M and GeForce 7300M were used.

Filter	Evolved	Conventional
Dilation/Erosion	0.1306/0.4160	-/-
Edges/Shots	4.3896/2.2172	-/4.2166
Gaussian/Shots	10.3673/2.2676	10.0192/4.2166

Table 1: Mean error per pixel for a filter given by our algorithm and for a conventional filter [4]

Some combinations of filters yield better results than other ones. It is not known what filters to combine for the best results. All these aspects open many possibilities for further experiments and research.

The usage of polymorphic image filters would be beneficial when trying to save space on a chip, because both filters reuse the same computing elements as well as their interconnections.

Acknowledgements. The Author of this work would like to thank to his supervisor Prof. Ing. Lukáš Sekanina, PhD., for his help and support.

References

[1] V. Salajka. Polymorphic image filters. Master's thesis, Faculty of Information Technology, Brno University of Technology, 2011.

[2] L. Sekanina, R. Růžička, Z. Vašíček, R. Prokop, and L. Fujcik. Repomo32 - new reconfigurable polymorphic integrated circuit for adaptive hardware. In *Proc. of the 2009 IEEE Symposium Series on Computational Intelligence - Workshop on Evolvable and Adaptive Hardware*, pages 39–46. IEEE Computational Intelligence Society, 2009.

[3] L. Sekanina and V. Salajka. Towards new applications of multi-function logic: Image multi-filtering. In *Proc. of the Design, Automation and Test in Europe, DATE*, pages 824–827. EDAA, 2012.

[4] L. Sekanina, V. Salajka, and Z. Vašíček. Two-step evolution of polymorphic circuits for image multi-filtering. In *2012 IEEE World Congress on Computational Intelligence*, pages 432–439, 2012.

[5] L. Sekanina, Z. Vašíček, R. Růžička, M. Bidlo, J. Jaroš, and P. Švenda. *Evoluční hardware*. Academia, 2009.

[6] A. Stoica, R. Zebulum, X. Guo, D. Keymeulen, I. Ferguson, and V. Duong. Taking Evolutionary Circuit Design From Experimentation to Implementation: Some Useful Techniques and a Silicon Demonstration. *IEE Proc.-Comp. Digit. Tech.*, 151(4):295–300, 2004.

[7] S. Tanachutiwat, J. U. Lee, W. Wang, and C. Y. Sung. Reconfigurable multi-function logic based on graphene p-n junctions. In *Design Automation Conference, DAC*, pages 883–888. ACM, 2010.