Functional-Level Development of Image Filters by Means of Cellular Automata

Michal Bidlo
Brno university of Technology
Faculty of Information Technology
IT4Innovations Centre of Excellence
Božetěchova 2
61266 Brno, Czech Republic
Email: bidlom@fit.vutbr.cz

Zdenek Vasicek
Brno university of Technology
Faculty of Information Technology
IT4Innovations Centre of Excellence
Božetěchova 2
61266 Brno, Czech Republic
Email: vasicek@fit.vutbr.cz

***Abstract*—A developmental method based on one-dimensional uniform cellular automaton is presented for generating image filters at the level of functional blocks. The key idea is to enhance the local transition function of cellular automaton in order to enable its cells to generate functional blocks when determining the new states during development. Simple genetic algorithm is applied to find a suitable cellular automaton (its initial state and the transition function) that is able in a finite number of steps to generate a functional structure for image filtering. Several sets of experiments are presented considering various settings of parameters of the developmental system. The evolved filters are evaluated using different types of grayscale images corrupted by salt-and-pepper noise of various intensity. The obtained filters are compared to some conventional median filters with respect to the filtering quality.**

I. Introduction

Evolutionary design represents a wide research area in which non-trivial problems are solved using evolutionary algorithms. Various techniques have been applied so far in different areas (e.g. mechanical constructions, functional structures, analog and digital circuits and so on). One of the key issue for successful evolutionary design is the representation problem, i.e. how the candidate solutions (called phenotypes) are encoded in evolutionary algorithms. For example, Miller's Cartesian Genetic Programming (CGP) [1] allows us to encode graph-based structures (e.g. digital circuits or neural networks) into linear strings whose information can directly be mapped onto a candidate solution. However, the direct mapping may be limiting for some application. For instance, Developmental CGP was demostrated to be more powerful technique in case of evolutionary design of large parity circuits [2]. Another example of a successful application of indirect encoding represents evolutionary design of generic sorting networks using Instruction-Based Development [3]. Development can also control the construction of a target object according to external conditions [4].

The developmental approach was originally inspired by a biological process of ontogeny. The ontogeny represents a developmental process involving formation of a multicellular organism from a zygote. It is influenced by genetic information of the organism and the environment in which the development is carried out. Computational development is usually considered as a non-trivial and indirect mapping from genotypes to phenotypes in an evolutionary algorithm to provide a more flexibility in the construction process of a candidate solution than that is achievable by direct mappings. In such case the genotype has to contain instructions (an algorithm) for the construction of a target object. While the genetic operators work with the genotypes, the fitness calculation (evaluation of the candidate solutions) is applied on phenotypes created by means of the development. The principles of the computational development together with a brief biological background and selected application of this bio-inspired approach are summarized in [5]. There are several approaches of how to perform computational development, for example Miller's Developmental Cartesian Genetic Programming [2], Koza's Developmental Genetic Programming [6], Instruction-Based Development [3] or cellular automata [7].

Several approaches have been published involving cellular automata as a developmental model in order to solve a specific task. Evolutionary algorithms have usually been applied to discover a suitable transition function of the cellular automaton. For example, Miller investigated the problem of evolving a developmental program inside a cell to create multicellular organism of an arbitrary size and characteristic [8]. He presented a system in which the organism organizes itself into well defined patterns of differentiated cell types (e.g. the French Flag). Tufte and Haddow utilized a FPGA-based platform of Sblocks [9] for the online evolution of digital circuits. The system actually implements a cellular automaton whose development determines the functions and interconnection of the Sblock cells in order to realize a function [10]. The evolutionary algorithm was utilized to design the rules for the development of the CA [10]. In recent years cellular automata were successfully applied in the task of circuit development at the gate level. In this case genetic algorithms were applied in order to design transition functions that are able to generate various types of digital circuits, e.g. polymorphic circuits [11], combinational multipliers [12], dividers [13] or sorting networks [11], [14]. In addition, advanced representations of transition function has also been studied [15], [14].

Cellular platforms have also been applied in the area of image processing. For example, an asynchronous 2D cellular automaton was evolved by Slatnia et al. in order to perform edge-detection task [16]. Similar problem was solved in [17], where an evolvable-hardware approach was considered to implement a highly parallel platform for low-level image processing. The problem of classification of satellite images using cellular automata was studied in [18]. The authors applied an approach based on the relationship between spectral reflectances and physical attributes of the image. In [19] Paul et al. proposed a cellular automata-based transform coding for image compression.

In this paper we will continue in the research devoted to the circuit design by means of cellular automata. The goal is to utilize one-dimensional uniform cellular automata for the development of image filters whose purpose is to remove a specific kind of noise from a corrupted image. A genetic algorithm will be applied to design a transition function that is able to generate the target filter in a finite number of steps of the cellular automaton. In contrast with the aforementioned approaches, we will consider functional-level development rather than utilizing basic logic gates. Hence the building blocks of the target filters implement more complex functions which perform operations over integer values representing pixel shades of the image to be filtered. It will be shown that cellular automata can be designed by evolution that are able to generate functional structures for image filtering. The obtained results will be analysed with respect to the output quality of the filtered images. A comparison with some existing conventional approaches will also be performed.

II. Cellular Automata

Cellular automata are discrete dynamical systems in which cells are organised in a regular structure. In case of one-dimensional (1D) cellular automata, that are considered in this paper, the cells constitute a finite linear structure. Each cell may occur in one state from a finite set of states. The states are updated synchronously in parallel in discrete time steps according to a local transition function. An update of all cells of a CA at a given time is referred to as a developmental step. The next state of a cell depends on the combination of states in the cellular neighborhood. In this paper we will consider the cellular neighborhood consisting of a given cell and its two immediate neighbors. Moreover, cyclic boundary conditions will be considered which means that the left-most and right-most cell of the CA are considered to be neighbors. Therefore, the cellular structure may be viewed as a circle. The local transition function defines a next state for every possible combination of states in the cellular neighborhood. Let us denote $c_1c_2c_3 \rightarrow c_n$ as a rule of the local transition function, where $c_1c_2c_3$ represents the combination of states of the cells in the cellular neighborhood and c_n denotes the next state of the cell to be updated. Then the cell in the middle of the neighborhood is updated from state c_2 to state c_n after a developmental step. In case of uniform cellular automaton the local transition function is identical for all the cells.

III. Developmental Model for Image Filters

In this section a developmental model will be described that allows us to generate image filters by means of uniform 1D cellular automata. The main idea of the CA-based structural development is based on a suitable enhancement of local transition function that enables the cells to generate functional blocks during the CA development. In this paper the goal of the developmental process is to design a functional structure (image filter) that is able to calculate a filtered value using the information contained in the local neighborhood of a given pixel (so-called filter window). The filter will operate over 8-bit grayscale images. Hence we have proposed to enhance each rule of the transition function to include, in addition to the next state, an information specifying a functional block and connection of its inputs to be generated by a given cell when the cell determines its next state. The following form of a rule of local transition function will be considered.

$$c_1c_2c_3 \rightarrow c_n : f\ i_1\ i_2$$

The part on the right of the colon specifies the functional block to be generated, i.e. its function (f) and the indices of two inputs (i_1, i_2) determining the connection of the block. Since the update of cell states in a developmental step is performed in parallel for all cells, one level of a target circuit is generated in a single developmental step of the CA.

In case of the first step the inputs of the blocks being generated are connected to the primary inputs of the target filter (i.e. to specific pixels of the filtering window). In the subsequent developmental steps the inputs of the functional blocks are connected to the outputs of blocks generated in the previous developmental step. After the last developmental step the output of the block generated by the central cell is considered as the output of the filter.

In order to enable the aforementioned possibilities of input connections, i_1 and i_2 are encoded as integers from which the specific index in the appropriate range is calculated using the modulo division. Specifically, in order to determine the connection of a block's input that is generated in the first developmental step, the values of i_1 and i_2 are divided by the value 9 (i.e. the number of pixels in the filter window). For the connection of inputs of blocks generated in subsequent steps, the value W is applied in the division, where W equals the number of CA cells.

In order to illustrate the proposed developmental model, let us consider an example of 3-cell binary cellular automaton for generating a filter that works with 3x3-pixel window (Figure 1a). Let an initial CA state be 100 and let a specific rules of local transition function be given (Figure 1b). The resulting filter is generated as follows. The top cell of the CA calculates its next state according to rule II, hence the state of this cell during the first developmental step is 1 and a block performing function 2 is generated whose inputs are connected to pixels in the filter window specified by the last two integers of that rule, i.e. the first input is connected to pixel $203\ mod\ 9 = 5$ and the second input to pixel $253\ mod\ 9 = 1$. The middle

TABLE I
SET OF FUNCTIONS REPRESENTING THE BUILDING BLOCKS OF IMAGE FILTERS

Code	Function	Description
0	in_1	value from input 1
1	in_2	value from input 2
2	$min(in_1, in_2)$	minimum
3	$max(in_1, in_2)$	maximum
4	$255 - in_1$	inversion
5	$in_1/1$	division by 2
6	$in_2/2$	division by 4
7	$in_1 + in_2$	addition
8	$in_1 +^S in_2$	addition with saturation
9	$(in_1 + in_2)/2$	average value
10	$\|in_1 - in_2\|$	absolute difference

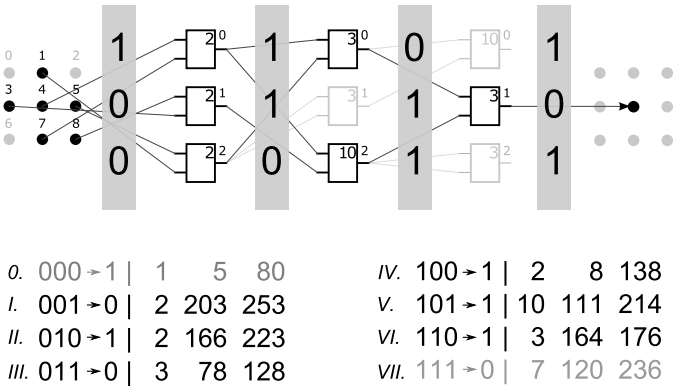


Fig. 1. Example of image filter development using 3 steps of cellular automaton.

cell gains its new state 1 according to rule IV and generates function 2 with its inputs connected to pixels $8\ mod\ 9 = 8$ and $138\ mod\ 9 = 3$. The same approach is utilized to calculate next state of the bottom cell.

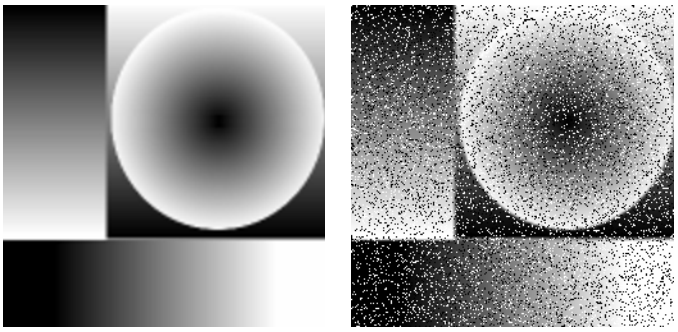
The second developmental step differs from the first step in the mapping of the block's inputs that are only allowed to connect to the outputs of blocks generated in the previous developmental step. The top cell determines its next state 0 according to rule III and generates a block with function 3 with its inputs connected to the top block (index $78\ mod\ 3 = 0$) and bottom block (index $128\ mod\ 3 = 2$) generated in the previous step. Similarly, the middle cell in the second step applies rule VI giving the new state 1 and generating function 3 with both its inputs connected to index $164\ mod\ 3 = 176\ mod\ 3 = 2$. The same principle is applied to calculate the last step of CA after which the output of the middle block is considered as the filter output (i.e. the filtered value of pixel 4 in the filter window). Note that there are some rules — specifically rule O and rule VII — in the transition function marked in gray which means that those rules have not been applied during the 3-step development of CA. In the filter structure developed by the CA the gray functional blocks represent filter elements that have no effect on the filter output and hence they can be removed. In particular, it is a case of middle block from the second step and top and bottom block generated in the third step.

IV. EVOLUTIONARY DESIGN OF CELLULAR AUTOMATA FOR FILTER DEVELOPMENT

A simple genetic algorithm (GA) was utilized for the evolutionary design of cellular automata that are able to generate image filters at the functional level. Each chromosome of the GA encodes an initial CA state and its complete local transition function. The rules of the transition function are represented by a 4-tuples, each of which contains the next cell state, function of a block to be generated and indices of its two inputs. The index (position) of the rule in genome is specified implicitly by means of the value expressed by the number representing the combination of states in the cellular neighborhood. The base of this number equals the number of possible cell states. Therefore, if we consider the general form of the rule $c_1\ c_2\ c_3 \rightarrow c_n : f\ i_1\ i_2$, only the part on the right of the arrow is encoded in the genome. For example, if a cellular automaton with 2 cell states and the cellular neighborhood consisting of 3 cells ought to be evolved, there are in total 2^3 rules of the transition function. Consider rule $0\ 1\ 1 \rightarrow 0 : 2\ 0\ 1$. Since the combination of states $0\ 1\ 1$, according to which new cell state will be calculated, corresponds to binary representation of number 3, this rule will be placed in the chromosome at position 3 of the local transition function. Note that such rule will be encoded as a sequence of integers $0\ 2\ 0\ 1$.

The population consists of 50 chromosomes which are initialized randomly at the beginning of evolution. The chromosomes are selected by means of the tournament operator with the base 4. No crossover operator is applied. Considering the mutation, our previous experiments showed that if only one gene per chromosome is mutated, then the convergence of the evolution can be very slow. Therefore, more genes will be allowed to mutate as follows. In each chromosome 6 candidate genes are chosen randomly. Every candidate gene undergoes mutation with probability 0.96. If a gene ought to be mutated, then a new value of the gene is generated randomly for which replaces the original value. A gene is understood as a single value representing a cell state or function code or input index. The high mutation rate was chosen in order to enable a larger change in the genome because of omitting the crossover operator. A single evolutionary run is performed for 500 thousands of generations and after this period the best solution in the population is considered as a resulting filter.

The fitness evaluation of candidate solutions is performed as follows. A given number of steps is performed by cellular automaton according to its specification encoded in each chromosome. Since it is impossible to evaluate all the input test vectors in the fitness function (for example, there are in total 256^9 different combinations for 3x3-pixel window in 256 degrees of grayscale), it is necessary to use a training image. The obtained filter, that is generated by the CA development, is applied on a corrupted image I_c in order to produce a filtered image I_f whose quality is evaluated by comparing it to a reference image I_r (i.e. the uncorrupted version of I_c).



(a) reference image (b) corrupted image

Fig. 2. Training image used to evaluate candidate filters during evolution.

The training image that was used in this paper is an artificial image whose corrupted version considers 20% salt-and-pepper noise (Figure 2). The training vectors were generated as 9-tuples from the corrupted image and the corresponding correct values from the reference image. In particular every 8-bit pixel value of the filtered image is calculated using the value of the corresponding pixel in the corrupted image and its immediate neighbouring pixels (i.e. the filter works with 3x3-pixel window).

In order to evaluate the quality of the filter in fitness function, mean absolute error (MAE) will be considered as the only criterion. If the corrupted image is of the size $R \times C$ pixels and 3x3-pixel filter window is used, then the number of filtered pixels is equal to the $(R-2) \times (C-2)$ (the boundary pixels are not considered). Formally the fitness function can be represented by Equation 1.

$$fit(I_f, I_r) = \frac{1}{(R-2)(C-2)} \sum_{i=1}^{R-2} \sum_{j=1}^{C-2} |I_f(i,j) - I_r(i,j)| \,. \quad (1)$$

The goal of the evolutionary algorithm is to design a filter which minimizes the difference between I_f and I_r, i.e. the fitness function calculates absolute error of I_f with respect to I_r. Note that the training image data, represented byt the pair of images I_c and I_r, are used to determine the quality of the candidate filters during the evolutionary design process. In order to evaluate the robustness of the resulting (evolved) filters, a set of various test images (corrupted by a given type of noise) is considered as will be described in Section V-B.

V. Experimental Results and Discussion

The goal of the experiments was to find a CA that is able to generate image filter for salt-and-pepper noise using building blocks from Table I. Several sets of experiments were conducted considering various settings of the cellular automaton, i.e. the number of cells, cell states and developmental steps. The ranges of these parameters were determined experimentally according to analysis of the filtering quality of resulting filters and time requirements of evolutionary experiments. For example, we determined that only 2 cell states do not provide sufficient scope for generating filters with reasonable filtering quality. Therefore, 3 states were identified as the minimal number of states. For each setup 100 independent evolutionary runs were performed.

TABLE II
Statistical results of the development of image filters using cellular automata. The block count is presented for the best and worst solutions with respect to their fitness.

CA configuration			Fitness value (MAE)			Block cnt.	
cells	states	steps	best	worst	mean	best	worst
9	3	5	1.787	4.498	2.539 ± 0.51	18	12
9	3	7	1.612	4.661	2.654 ± 0.55	34	10
9	3	9	1.682	4.393	2.708 ± 0.55	44	36
9	4	5	1.536	3.791	2.124 ± 0.41	23	10
9	4	7	1.418	4.693	2.372 ± 0.66	26	15
9	4	9	1.534	4.961	2.591 ± 0.67	34	34
9	8	5	1.306	6.243	2.460 ± 1.00	20	14
9	**8**	**7**	**0.926**	6.813	2.229 ± 1.07	35	24
9	**8**	**9**	**0.939**	13.416	3.014 ± 1.94	24	42
7	3	5	1.695	5.167	2.482 ± 0.54	16	10
7	3	7	1.581	4.581	2.502 ± 0.55	26	21
7	3	9	1.698	4.285	2.698 ± 0.60	26	27
7	4	5	1.485	4.091	2.044 ± 0.37	20	12
7	4	7	1.177	4.004	2.181 ± 0.61	19	20
7	4	9	1.235	5.949	2.322 ± 0.63	42	38
7	8	5	1.079	5.431	2.397 ± 0.91	14	11
7	***8***	***7***	***0.756***	10.194	2.696 ± 1.63	28	31
7	**8**	**9**	**0.911**	14.675	2.906 ± 2.11	36	36
5	3	5	1.491	4.979	2.758 ± 0.55	15	16
5	3	7	1.771	4.441	2.742 ± 0.57	27	22
5	3	9	1.828	5.373	2.924 ± 0.67	30	27
5	4	5	1.339	6.203	2.385 ± 0.64	16	15
5	4	7	1.235	4.487	2.446 ± 0.60	23	21
5	4	9	1.470	6.101	2.707 ± 0.75	27	21
5	8	5	1.355	6.970	2.875 ± 0.92	15	17
5	**8**	**7**	**1.024**	10.142	3.165 ± 1.64	19	11
5	**8**	**9**	**1.191**	15.652	3.549 ± 2.31	30	22

Statistical results of the experiments are summarized in Table II. The best filter was identified in every set of results considering mean average error (MAE) of the filtered image with respect to the reference image as the only criterion.

As evident, the fitness values do not exhibit any noticeable dependence on the CA parameters. Both the best fitness and mean values mostly have only small differences considering various setups. Greater differences can be observed for higher number of states which is probably caused by the fact that the search space becomes very huge and evolution may need longer time (i.e more generations of the GA) to find a reasonable solution. Despite of this fact, two fittest filters were discovered only for the setups with higher number of cell states and developmental steps (marked in **bold** in Table II). In fact, this observation may be expectable because there are more possibilities of how to map the functional blocks on the rules of transition function for higher number of states. Nevertheless, it is not generally true that the more cells the better filters can be found because the best filters from all the

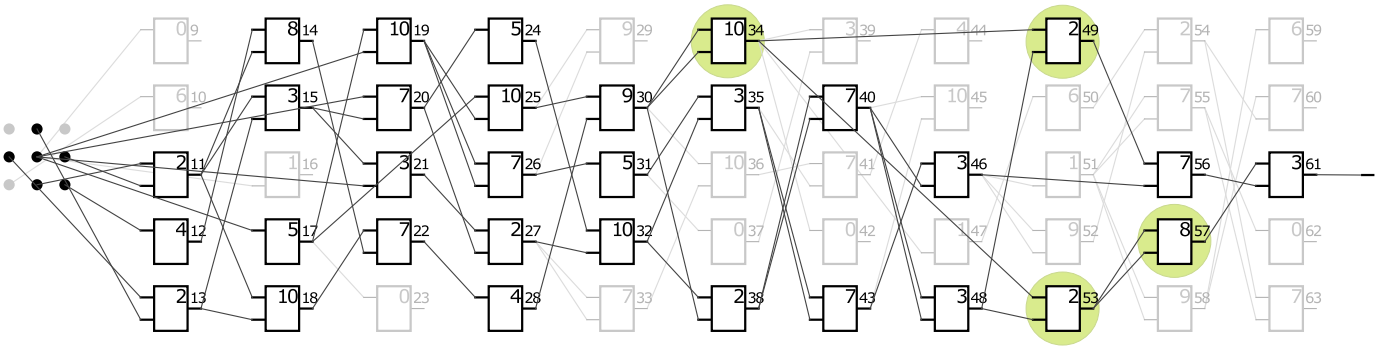


Fig. 3. Example of image filter generated by means of cellular automaton whose function is preserved for a limited number of developmental steps that were not considered during evolution. The blocks whose output will permanently be 0 are marked by shaded circles.

sets of experiments were developed for 7 cells which is the median of the number of cells considered in the experiments. The setup that generated the fittest filter, that was discovered in this paper, is marked in ***bold italic*** in Table II. This table also contains information about the number of effective functional blocks of the best filter found in each specific setup. It can be seen that the number of blocks varies substantially (from 14 to 44) which is mostly dependent on the number of cells and steps. What may be interesting, however, is the fact that most of the best filters consist of about mean value of the aforementioned range.

A. Filters Generated by the Evolved Cellular Automata

The structure of the best evolved filter is depicted in Figure 4. This filter consists of 30 blocks and utilizes the whole filter window to calculate the filtered value. We evaluated the CA discovered for the construction of this filter also for more developmental steps and investigated the filtering properties of the resulting functional structures. Unfortunately, no other reasonable filtering function was observed. If more levels of the filter are generated using this CA, then its function is degraded and filtering quality decreases substantially.

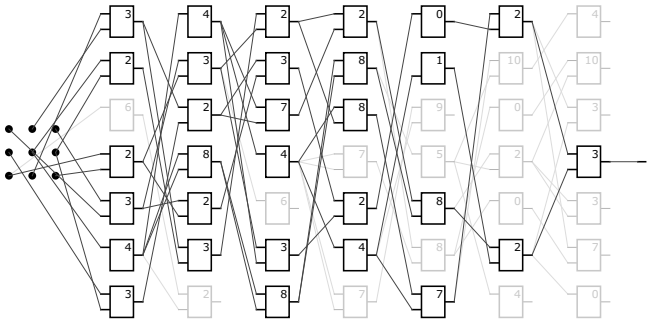


Fig. 4. The best filter evolved from all the experiments.

It was shown that although the evaluation of the filters during evolution is performed after a given number of developmental steps, the filtering function in some cases provides reasonable quality even if the development continues. For example, a CA was discovered that exhibits the ability to preserve filtering quality of the generated functional structure obtained by some additional developemntal steps. The resulting filter circuit is depicted in Figure 3. There were 9 developmental steps of the CA performed during evolution after which the filter function was evaluated. The filter output is connected to the output of the block with index 51. Subsequent analysis showed that the same function has already been observable after the eighth developmental step (i.e. at output 46). Moreover, if the development of evolved CA continues, the generated circuit is able to preserve its function also for the 10th (output 56) and 11th (output 61) developmental step (i.e. some steps that were not considered during evolution). This feature is based on the fact that the result of function 10 (absolute difference) at output 34 will always provide 0 because its inputs are connected to the same block. This result is propagated through the identity function blocks (outputs 42 and 47). This zero at output 47 is then connected in the 9th steps to blocks of function 2 (minimum) which ensures that 0 is propagated as minimal value at outputs 49 and 53. In the 10th step the output 56 of functional block 7 (addition) will always possess the value of function from output 51 which preserves the filtering quality also in the 10th step. Finally, zero is propagated through block 8 (addition with saturation) at output 57 and because block 3 (maximum), generated in the 11th step, takes as its first input the zero from output 57 and as its second input the filtered value from output 56, its result at output 61 will again provide the filtered value that represents the results of the evolved filter function. In fact, blocks with outputs 49, 53, 56, 57 and 61 only enables the filtered value from output 46 to be propagated through the circuit structure generated in the 10th and 11th step. It was determined that by subsequent development of the CA (i.e. the 12th, 13th,... step) the evolved function is destroyed because of generating unsuitable block functions.

Figure 5 shows an example of filter created by means of another evolved cellular automaton. The developed functional structure was evaluated during evolution after the ninth step of the CA. Similarly to the example described in the previous paragraph, this CA is also able to produce filters with the same fitness values using less number of developmental steps. However, its exact function seems to be much more difficult

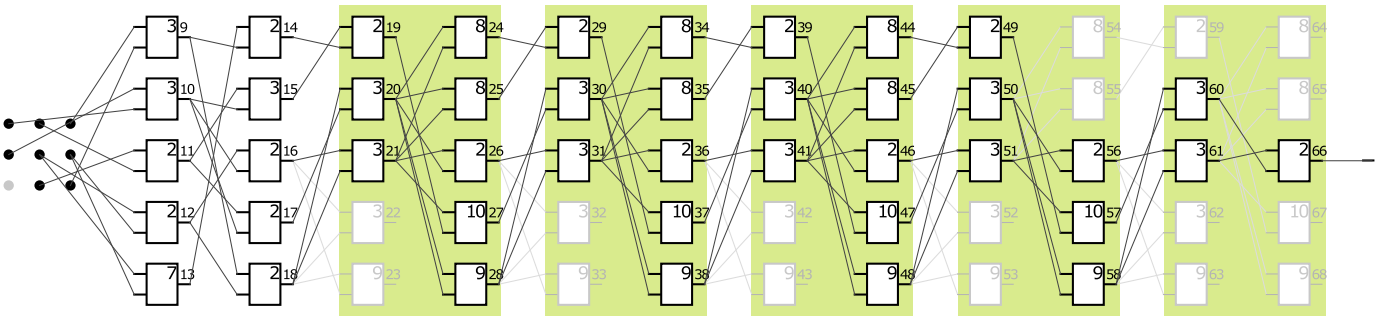


Fig. 5. Example of image filter developed by means of cellular automaton whose function is preserved for arbitrary number of developmental steps that were not considered during evolution. The periodically repeated structure, that is generated by the CA, is marked by shaded rectangles.

to analyze. As evident from Figure 5, the CA generates in the third and fourth developmental step a two-level block structure that is repeating periodically during the subsequent steps. If the fitness value of filters generated for several developmental steps is analyzed, we can observe decreasing fitness values. The reasonable fitness value is obtained after the seventh step (1.8625). This value is then slightly improved in the ninth step to 1.8196. After that, the value no longer changes with subsequent development of the CA. It demonstrates that the CA is able to periodically generate a circuit structure whose function is able in the first phase to optimize the filtered value and in the second phase to preserve the optimized value during subsequent development.

B. Evaluation of the Filtering Quality

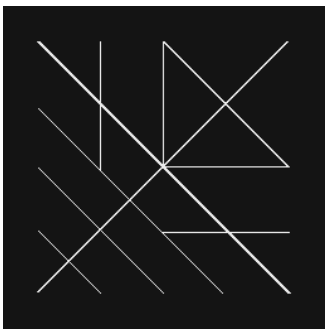
In order to determine the filtering properties of the evolved filters, we have used a set of common test images (airplane, baboon, barbara, bird, bridge, camera, crosses, goldhill, lena, peppers, squares) consisting of 256×256 pixels corrupted by various intensity of the salt-and-pepper noise. Table III shows average MAE values for the fittest five evolved filters and some conventional filters. We have used common median filter (denoted as med$N$) and advanced method known as adaptive median filter (denoted as amed$N$). It can be seen that the evolved filters are in most cases better or at least of the same quality in comparison with the conventional solutions utilized to suppress the salt-and-pepper noise. Very good results of the evolved filters are observed not only for higher but also for lover noise intensities. Note that the filters were trained for 20% salt-and-pepper noise. The problem of common median filter is that filtering images tend to be smudged. Moreover, the larger filter window the worse image quality. This issue is observable in Table III, where the average MAE values of the median filters represent the worst results. On the contrary, the evolved filters are able to remove substantial part of noise while preserving image details. This fact is clearly demonstrated by the presence of the best MAE values for lower noise intensity (up to 10%).

The filtering capability of the best evolved filter using lower noise intensity (10%) is illustrated at Figure 6. This example actually represents a special case of a test image used to evaluate the detail preserving capabilities of the image filters.

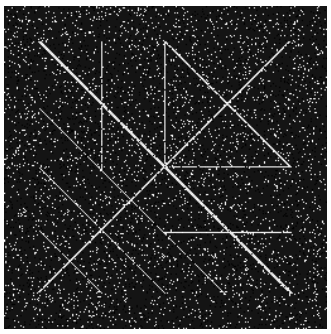
TABLE III
AVERAGE MAE VALUES FOR THE BEST FIVE EVOLVED FILTERS EVALUATED ON A SET OF TEST IMAGES IN COMPARISON WITH SOME CONVENTIONAL MEDIAN FILTERS. THE NUMBER OF CELLS (W), CELL STATES (S) AND STEPS (T) ARE UTILIZED FOR UNIQUE IDENTIFICATION OF EVOLVED FILTERS. THE CONVENTIONAL FILTERS CONSIDERED FOR THE EVALUATION INCLUDE STANDARD MEDIAN FILTER WITH 3X3 WINDOW (MED9) AND 5X5 WINDOW (MED25) AND ADAPTIVE MEDIAN FILTER WITH 3X3 WINDOW (AMED9) AND 5X5 WINDOW (AMED25).

	Noise level				
Filter	1%	5%	10%	15%	20%
W7S8T7 (0.756)	0.108	0.345	0.880	1.572	2.476
med9	3.557	3.761	4.030	4.369	4.855
med25	5.670	5.774	5.903	6.043	6.219
amed9	1.313	1.388	1.529	1.760	2.179
amed25	1.574	1.525	1.583	1.703	1.943
W7S8T9 (0.911)	0.589	0.474	1.148	1.954	2.938
med9	3.557	3.761	4.030	4.369	4.855
med25	5.670	5.774	5.903	6.043	6.219
amed9	1.313	1.388	1.529	1.760	2.179
amed25	1.574	1.525	1.583	1.703	1.943
W9S8T7_1 (0.926)	0.148	0.343	0.875	1.587	2.594
med9	3.557	3.761	4.030	4.369	4.855
med25	5.670	5.774	5.903	6.043	6.219
amed9	1.313	1.388	1.529	1.760	2.179
amed25	1.574	1.525	1.583	1.703	1.943
W9S8T9 (0.939)	0.148	0.400	1.015	1.823	2.957
med9	3.557	3.761	4.030	4.369	4.855
med25	5.670	5.774	5.903	6.043	6.219
amed9	1.313	1.388	1.529	1.760	2.179
amed25	1.574	1.525	1.583	1.703	1.943
W9S8T7_2 (0.971)	0.058	0.343	0.896	1.677	2.795
med9	3.557	3.761	4.030	4.369	4.855
med25	5.670	5.774	5.903	6.043	6.219
amed9	1.313	1.388	1.529	1.760	2.179
amed25	1.574	1.525	1.583	1.703	1.943

In this case it is possible to emphasize the robustness of the evolved filter in comparison with the conventional solutions. As evident from Figure 6f, the evolved filter was able to nearly completely remove 10% noise while preserving sharp thin lines in the filtered image. On the contrary, the median filters failed because substantial part of the image was destroyed.

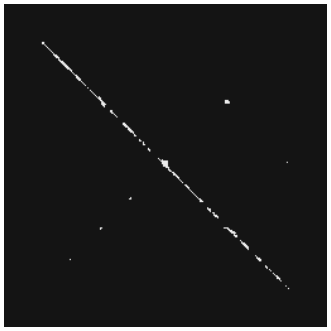


(a) original image



(b) corrupted image

(c) median filter 3x3



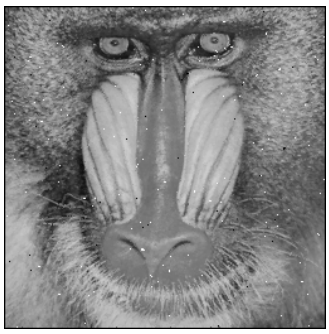
(d) median filter 5x5

(e) adaptive median filter 3x3

(f) evolved filter 3x3

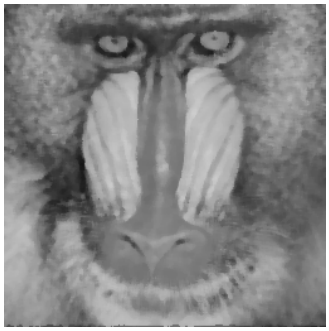
Fig. 6. Filtering a test image (crosses) corrupted by 10% salt-and-pepper noise using the best evolved filter W7S8T7 and selected conventional median filters.

(a) corrupted image

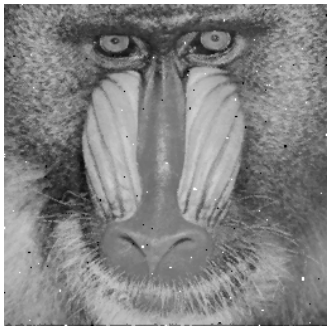


(b) evolved filter 3x3

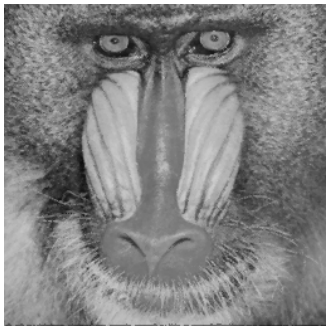
(c) median filter 3x3



(d) median filter 5x5



(e) adaptive median filter 3x3



(f) adaptive median filter 5x5

Fig. 7. Filtering a test image (baboon) corrupted by 20% salt-and-pepper noise using the best evolved filter W7S8T7 and selected conventional median filters.

Figure 7 demonstrates the filtering quality of the best evolved filter for high noise intesity (20%). As expected, the adaptive median filter (Figure 7f), designed to provide resonable results while preserving image details, provides the best result. However, the evolved filter (Figure 7b) also provides very good result that is comparable to adaptive median filter utilizing the same filter window (i.e. 3x3 pixels). Whilst minor part of noisy pixels remained, closer look at Figure 7b may indicate that the evolved filter is able to provide better contrast and level of detail. Although all the noisy pixels were removed by common median filters, these filters failed with respect to their output quality because the filtered images exhibit a noticeable loss of details.

VI. Conclusions

A developmental model based on uniform 1D cellular automaton was presented for generating image filters at the level of functional blocks. In fact, it is the first case when cellular automata were applied for generating circuit structures in the area of image processing. Simple genetic algorithm was utilized in order to find a suitable cellular automaton (i.e. its initial state and local transition function) that is able in a finite number of steps to generate a functional structure that exhibit a reasonable filtering quality. It was determined that although the evaluation of the filters during evolution is performed after a given number of developmental steps, the filtering function in some cases provides reasonable quality already after lower number of steps which was not explicitly required during evolution. This feature enables to reduce the

number of functional blocks as well as delay of the filters after the evolutionary process. Moreover, some cellular automata demonstrated an ability to preserve the filtering function if the CA development continues.

The evolved filters were evaluated on a set of test images corrupted by salt-and-pepper noise of various intensity and their quality was compared to some conventional median filters that are usually applied to eliminate this type of noise. It has been shown that the filters developed by mean of cellular automata exhibit in many cases the best results in comparison with the conventional filters. We have demonstrated that the best evolved filter is able to remove substantial part of noise even for higher intensity (20%), to preserve details of the filtered image and provide exceptional result when filtering special (nearly black-and-white) images in which case the median filters failed.

In general, the proposed approach has shown as a promising developmental model for the design of a wide range of functional structures (multipliers, dividers, image filters and many others. In case of using uniform CA, it is needed to design a transition function that is common for all cells. It means that the process of creating the target structure *emerges* from the CA development which is a result of inherent local interactions between the cells. This approach may be especially beneficial for tasks whose organization of building blocks is not strictly defined or whose resuts may not be exact by nature (e.g. image processing or so-called approximate computing). However, the problem of utilizing a specific CA setup (the number of states, interpretation of the transition rules with respect to additional information that need to be evolved) seems to be a challenging task whose solution is mainly a subject of experimental work.

The CA-based developmental model was, in fact, considered with basic settings which include the following. The maximal number of cells was limited to the number of circuit inputs considering 3x3 filter window, i.e. there were in total 9 input values. The development of CA generated a single level of filter structure per a developmental step, the inputs on functional blocks were allowed to connect to the filter inputs (only in case of the first step of CA) or to outputs of blocks generated during the immediately previous step. These parameters may be modified in order to extend the evolutionary search space and potentially increase the quality of evolved solutions. Therefore, these issues will constitute a basis for our future research in which different application domains will be considered.

ACKNOWLEDGMENT

This work was supported by the Czech science foundation project P103/10/1517, the research programme MSM 0021630528, the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070 and the BUT projects FIT-S-11-1 and FIT-S-12-1.

REFERENCES

[1] J. F. Miller and P. Thomson, "Cartesian genetic programming," in *Proc. of the 3rd European Conference on Genetic Porgramming, Lecture Notes in Computer Science, vol 1802*. Berlin Heidelberg New York: Springer, 2000, pp. 121–132.

[2] ——, "A developmental method for growing graphs and circuits," in *Proc. of the 5th Conf. on Evolvable Systems: From Biology to Hardware (ICES 2003), Lecture Notes in Computer Science, vol. 2606*. Berlin DE: Springer-Verlag, 2003, pp. 93–104.

[3] M. Bidlo and J. Škarvada, "Instruction-based development: From evolution to generic structures of digital circuits," *International Journal of Knowledge-Based and Intelligent Engineering Systems*, vol. 12, no. 3, pp. 221–236, 2008.

[4] M. Bidlo and L. Sekanina, "On impact of environment on the complexity generated by evolutionary development," in *MENDEL 2010 - 16th International Conference on Soft Computing*. Faculty of Mechanical Engineering BUT, 2010, pp. 501–508.

[5] S. Kumar and P. J. Bentley (eds.), *On Growth, Form and Computers*. Elsevier Academic Press, 2003.

[6] J. R. Koza and M. A. Keane and M. J. Streeter and W. Mydlowec and J. Yu and G. Lanza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003.

[7] J. von Neumann, *The Theory of Self-Reproducing Automata*. A. W. Burks (ed.), University of Illinois Press, 1966.

[8] J. F. Miller, "Evolving developmental programs for adaptation, morphogenesis and self-repair," in *Advances in Artificial Life. 7th European Conference on Artificial Life, Lecture Notes in Artificial Intelligence, volume 2801*. Dortmund DE: Springer, 2003, pp. 256–265.

[9] P. C. Haddow and G. Tufte, "Bridging the genotype–phenotype mapping for digital FPGAs," in *Proc. of the 3rd NASA/DoD Workshop on Evolvable Hardware*. Los Alamitos, CA, US: IEEE Computer Society, 2001, pp. 109–115.

[10] G. Tufte and P. C. Haddow, "Towards development on a silicon-based cellular computing machine," *Natural Computing*, vol. 4, no. 4, pp. 387–416, 2005.

[11] M. Bidlo, Z. Vasicek, and K. Slany, "Sorting network development using cellular automata," in *Proc. of the 9th International Conference on Evolvable Systems: From Biology to Hardware (ICES 2010), Lecture Notes in Computer Science, vol. 6274*. Berlin Heidelberg New York: Springer, 2010, pp. 85–96.

[12] M. Bidlo and Z. Vasicek, "Investigating gate-level evolutionary development of combinational multipliers using enhanced cellular automata-based model," in *Proc. of The 2009 IEEE Congress on Evolutionary Computatio, CEC 2009*. IEEE Computer Society, 2009.

[13] ——, "Comparison of the uniform and non-uniform cellular automata-based approach to the development of combinational circuits," in *Proc. of The 4th NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2009*. IEEE Computer Society, 2009, pp. 423–430.

[14] ——, "Cellular automaton as a sorting network generator using instruction-based development," in *Cellular Automata for Research and Industry*, ser. Lecture Notes in Computer Science Volume 7495. Springer Verlag, 2012, pp. 214–223.

[15] ——, "Instruction-based development of cellular automata," in *Proc. of The 2012 IEEE Congress on Evolutionary Computatio, CEC 2012*. IEEE Computer Society, 2012.

[16] S. Slatnia, M. Batouche, and K. E. Melkemi, "Evolutionary cellular automata based-approach for edge detection," in *Proceedings of the 7th international workshop on Fuzzy Logic and Applications: Applications of Fuzzy Sets Theory*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 404–411.

[17] G. Hollingworth, A. Tyrrell, and S. Smith, "Simulation of evolvable hardware to solve low level image processing tasks," in *In Proc. of the Evolutionary Image Analysis, Signal Processing and Telecommunications Workshop, Lecture Notes in Computer Science Volume 1596*. Springer-Verlag, 1999, pp. 46–58.

[18] M. Espnola, R. Ayala, S. Leguizamn, and M. Menenti, "Classification of satellite images using the cellular automata approach." in *World Summit on the Knowledge Society, WSKS 2008*, ser. Communications in Computer and Information Science, vol. 19. Springer, 2008, pp. 521–526.

[19] K. Paul, D. R. Choudhury, and P. P. Chaudhuri, "Cellular automata based transform coding for image compression," in *Proceedings of the 6th International Conference on High Performance Computing*, ser. HiPC '99. London, UK: Springer-Verlag, 1999, pp. 269–273.