Implicit Fitness Sharing for Evolutionary Synthesis of License Plate Detectors

Krzysztof Krawiec and Mateusz Nawrocki

Institute of Computing Science, Poznan University of Technology,
Piotrowo 2, 60965 Poznań, Poland

Abstract. A genetic programming algorithm for synthesis of object detection systems is proposed and applied to the task of license plate recognition in uncontrolled lighting conditions. The method evolves solutions represented as data flows of high-level parametric image operators. In an extended variant, the algorithm employs implicit fitness sharing, which allows identifying the particularly difficult training examples and focusing the training process on them. The experiment, involving heterogeneous video sequences acquired in diverse conditions, demonstrates that implicit fitness sharing substantially improves the predictive performance of evolved detection systems, providing maximum recognition accuracy achievable for the considered setup and training data.

Keywords: Genetic programming, pattern recognition, image analysis, implicit fitness sharing, license plate recognition.

1 Introduction

Manual design of image analysis systems is a time-consuming task that requires a lot of expertise. Even for a skilled expert, the final outcome of a chain of image processing algorithms is hard to predict, so usually many designs have to be laboriously tested to come up with a well performing image analysis system.

In this study we automate this search process using genetic programming [7], allowing the search algorithm to compose complete image analysis programs. The programs maintained in the population are composed of instructions that implement image processing algorithms known from literature. The instructions are allowed to have parameters, which also undergo evolutionary tuning, so overall the method performs search in a joint space of *structures* (data flows) and *parameters* of image analysis programs.

The major contribution of this paper are the experimental outcomes that assess the method with respect to its ability to model the dependencies observable in data (training set performance) and capability of generalization (testing set performance). In particular, we propose to extend the basic approach with implicit fitness sharing, which entices the individuals in population to pay more attention to the particularly difficult examples.

2 Related Work

License plate recognition is one of the best established applications and a common benchmark for pattern recognition and computer vision systems. Former research on this topic engaged various paradigms of computational intelligence, including artificial neural networks, fuzzy logic, and evolutionary computation (see review in [10]). For instance, in [18], a fuzzy logic approach has been applied to this task. In [17], the authors use immune and genetic algorithms to acquire the parameters for the initial step of plate recognition. [15] uses genetic algorithm to optimize weights of neural network that performs the recognition task. In [6], genetic algorithm is used to determine the location of license plate in an image. Other examples of plate recognition systems involving techniques characteristic to computational intelligence can be found in, among others, [1,5,16].

Many of license plate recognition systems successfully implemented in real-world environments assume that vehicles are close to the camera, do not move (or are close to still), and the lighting is at least partially controlled (e.g., infrared emitters or flash light is involved). This is characteristic for deployments like authorization of entry for parking lots and gated blocks-of-flats. However, plate recognition task becomes much more challenging when performed in uncontrolled conditions, which is the case in this study, where the operating conditions resemble more CCTV (closed-circuit television) monitoring in urban areas. Most importantly, the camera used in the experimental part of this paper observes the moving vehicles from a relatively long distance. As a consequence, the observed projected dimensions of the plates are much smaller, and the images can be distorted by motion blur and perspective projection. Also, nothing is assumed about the lighting conditions. We allow also for the presence of multiple vehicles in the field of view.

3 The Approach

We divide the entire task of license plate recognition into four separate stages: motion segmentation, plate detection, character segmentation, and character recognition. Except for motion segmentation, each video frame is processed independently. The stage that undergoes evolutionary learning described in following is plate detection; the remaining stages have been designed manually and remain fixed during evolution. In particular, character recognition is carried out using a support vector machine (SVM, [3,13]), previously trained on a large collection of human-segmented characters belonging to 36 classes (26 uppercase Latin alphabet letters plus 10 digits). For the detailed description of the motion segmentation, character segmentation, and character recognition phases, see [8].

The task of plate detection stage is to determine, in a single frame (image), the locations of license plates, called *plate candidates* in following. To this aim, we employ tree-based genetic programming, with each individual (program) representing a complete license plate detector. Each program is a tree composed of instructions (nodes) implementing various image analysis algorithms that pass

Table 1. Image processing instructions employed by the method

Operator	Arguments	Description
thr	image i, float t	Thresholds i using threshold t
gamma	image i, float γ	Gamma correction of i
inv	image i	Inversion of i
exp	image i	Pixel-wise exponentiation of i
log	image i	Pixel-wise logarithm of i
$+, -, *, /$	image i, image j	Pixel-wise image arithmetic

the processed images to each other. An input image is fed into programs using the terminal nodes (leaves). The image produced at the root node is interpreted as the output of a program, where pixel brightness is assumed to represent program's confidence in the presence of license plate at a particular location. This image is postprocessed using a fixed (i.e., non-evolving) procedure, which involves thresholding and scanning for connected components. Every connected component found is replaced by a corresponding minimal bounding rectangle (MBR). If an MBR fulfills certain size and aspect ratio constraints, the image fragment enclosed by it is passed to the character segmentation stage (and, subsequently, character recognition stage).

Table 1 presents the set of instructions used to form the evolving programs. The full set of instructions embraces also terminal nodes, which include: R, G, B (red/green/blue channel of the input image), and H, S, I (hue/saturation/intensity channels). Finally, the So terminal provides the input image converted to grayscale and processed using the Sobel filter, while F terminal is the output of simple handcrafted plate detector, which we obtained using evolutionary tuning in our previous study [8].

The algorithm evolves a population of programs encoded in the way described above. In each generation, a new population of programs is bred using selection, mutation, and crossover operators. Selection is driven by the fitness values assigned to programs. In the standard variant, the fitness of an individual program s (candidate solution) is defined as the performance of the complete recognition system that uses s (i.e., composed of: plate detection implemented by s, character segmentation, and character recognition using the trained SVM), averaged over the training set of images T. Formally,

$$f_{std}(s) = \frac{1}{|T|} \sum_{t \in T} f(s,t) \tag{1}$$

where $f(s,t) \in [0,1]$ is the performance of s on example (image) t, based on the agreement of the character sequence recognized at the plate location indicated in t by s, and the true character sequence present in the license plate in frame t (see experimental part). Alternatively, we employ another fitness assessment method detailed in the next section.

4 Implicit Fitness Sharing

To learn an image analysis algorithm that robustly detects license plates, the training set should be diversified, embracing images of different cars, taken in various lighting conditions, from different aspects, at various visibility, etc. In such a diversified sample, some plates can be expected to be easier to detect and recognize than others. The ability to solve (recognize) the harder examples (plates) should be particularly appreciated during the learning process. An individual that acquires such capability at some stage of evolution should have greater odds for survival, even if it happens to fail on some easier examples. Unfortunately, the standard definition of fitness values all examples equally. As a result, the individuals that exhibit such unique skills may have problems to pass the selection stage.

Implicit fitness sharing (IFS), introduced by Smith *et al.* [14] and further explored for genetic programming by McKay [11,12], is a technique designed to overcome this deficiency. It weighs the reward granted for solving each example according to its difficulty, which is assessed based on how hard it appears to the individuals in the current population. Formally, the fitness f_{ifs} of an individual (candidate solution) s is defined as:

$$f_{ifs}(s) = \frac{1}{|T|} \sum_{t \in T(s)} \frac{1}{n(t)} \tag{2}$$

where $T(s) \subseteq T$ is the set of examples solved by s, and $n(t)$ is the number of individuals that solve t. Thus, the total amount of reward that any example t can pass onto individuals in population amounts to 1.0, and that amount is shared equally between the individuals that solve it. When all individuals in population P solve t, $n(t) = |P|$, and they all receive the same, minimal reward. If, on the other hand, t is solved by only one individual in P, $n(t) = 1$ and such an individual (and only it) will be granted the maximal reward of 1.0.

The fitness function defined by IFS entices individuals in population to solve examples that appear particularly difficult for the current state of the search process. Individuals that exhibit such unique capabilities are highly rewarded, which increases their odds for survival, and makes propagation of their traits to next generations more likely. In this respect, IFS may be seen as a diversity maintenance technique. Notably, it is also a rudimentary form of coevolution, as the fitness granted to an individual depends on the performance of the other individuals in the population.

Standard IFS assumes that an individual either solves an example or not. In the license plate recognition task, the performance on a single example (image) may vary gradually, depending on the number of correctly recognized plate characters, and is reflected by the function $f(s,t) \in [0,1]$ (cf. standard fitness definition in Eq. (1)). To take this into account, we redefine the IFS fitness in the following way:

$$f_{ifs}(s) = \sum_{t \in T} \frac{f(s,t)}{\sum_{s' \in P} f(s',t)} \tag{3}$$

Contrary to the standard IFS (Eq. 2), in this formula the expression in denominator calculates the total performance of all individuals in population P (including s) on example t (rather than *counting* the number of individuals that solve t). Nevertheless, the effect is analogous: performing well on an example that is hard to recognize/classify is more beneficial than doing so for an example that is deemed easy by the individuals in population.

5 The Experiment

The primary objective of the experiment was to assess the performance of the proposed approach and verify the usefulness of implicit fitness sharing. Thus, the following configurations have been considered: conventional genetic programming (GP) with standard fitness function (Eq. 1) and genetic programming driven by IFS (GP-IFS, Eq. 3).

The Data. The image data is a part of collection of 1233 frames of 160 different vehicles (mostly passenger cars) described in our former study [8]. The training and testing sets are disjoint and comprise, respectively, 97 and 98 images randomly selected from that database. Each frame has been manually inspected and the actual (true) license number has been assigned to it. Frames have been acquired using a stationary camera working with resolution 1280×960 pixels, located at 15-20 meters from the passing-by cars. The motion segmentation phase typically crops the frames to dimensions comparable to VGA standard, in which vehicles occupy on average 75% of the frame area. Most frames feature cars in frontal view. The plates to be recognized have typically dimensions of 150×30 pixels, however, they are often far from rectangular due to perspective projection and vehicle's tilt and yaw. The dataset has been acquired in realistic conditions and is highly heterogeneous: it comprises various lighting conditions (different time of the day, including backlight as well as plates directly exposed to sunlight), different whether conditions (both sunny and cloudy days), and with license plates subject to dirt and mounted at different heights relative to road level.

The Setup. For both GP and GP-IFS, we run generational evolution algorithm with a population of 100 individuals for 100 generations. Other parameters are set as follows: tournament selection with tournament size 7, tree-swapping mutation applied with probability 0.9, subtree-replacing mutation applied with probability 0.1. Evolutionary runs are repeated 10 times to lower the variance of performance. For the remaining parameters, we use the defaults of the ECJ package [9] that the evolutionary part of our framework is based on. The image analysis component employs the OpenCV library [2] written in C++. Communication between modules is facilitated via exchange of XML files.

The performance of individual s on example t has been defined as:

$$f(s,t) = \frac{d_{max} - \min\{d_{max}, d(s(t), act(t))\}}{d_{max}} \tag{4}$$

where $s(t)$ is the character string representing the plate number as read by s, $act(t)$ is the actual plate number present in t, and d is the Levenshtein distance

Table 2. Fitness (f_{std}) of the best-of-run individuals. Averages and medians over 10 evolutionary runs. Best-on-training is the test-set performance of the best of best-of-run individuals.

	Training set		Testing set		
Setup	Average	Median	Average	Median	Best-on-training
GP	0.890±0.033	0.897	0.572±0.210	0.681	0.745
GP-IFS	0.914±0.006	0.917	0.682±0.131	0.713	0.767

metric, i.e., the minimal number of insertions, deletions, and substitutions required to transform one character sequence into another. For the set of plates considered here, we set $d_{max} = 5$, so the maximal distance that positively contributes to fitness is 4 (most plate numbers used here had 7 characters). If $d(s(t), act(t)) \geq 5$, an individual scores 0 for the frame. For instance, this is the case when no plate candidate has been detected in a frame. For conventional fitness function (GP), individual's fitness is a normalized sum of $f(s,t)$ (see Eq.(1)) over all training examples. For GP-IFS, the fitness is Eq. (3).

In our previous study, we found out that discrimination of characters '0' (zero) and 'O' is extremely difficult for the SVM classifier. In the typeface used in the considered license plates, these characters differ only in aspect ratio, which can be easily distorted by perspective projection. Because good discrimination of these decision classes is impossible without, e.g., syntactic rules, we fuse them and treat these characters exchangeably.

The Results. In Table 2 we present the fitness of the best-of-run individuals for each setup, and their performance on a test set. Because f_{std} and f_{ifs} cannot be compared directly, the best-of-run individuals have been assessed using standard fitness f_{std}. GP-IFS clearly outperforms standard GP, particularly on the test set, which suggests that the use of implicit fitness sharing lowers the risk of overfitting. We verified this additionally by calculating the Pearson correlation coefficient between the training-set and testing-set performance over the 10 runs. For GP, no significant correlation was observed (0.03), while for GP-IFS, that correlation was strong (0.81).

The average is an unbiased estimator of the expected performance of a method, and as such allows meaningful comparison. However, a pragmatic human designer of a plate recognition system would not care much about these estimates; rather than that, he would look for the best performing program. To simulate this attitude, for each method, from the 10 best-of-run individuals, we selected also the individual that attained the highest fitness, and evaluated it on the testing set. The last column of Table 2 reports the outcomes of that evaluation. Also in this case, the IFS-based approach fares better.

The theoretical upper bound of f_{std} is 1.0. This however does not necessarily mean that perfect performance can be attained using a specific plate reader, by which we mean here the character segmentation algorithm and character

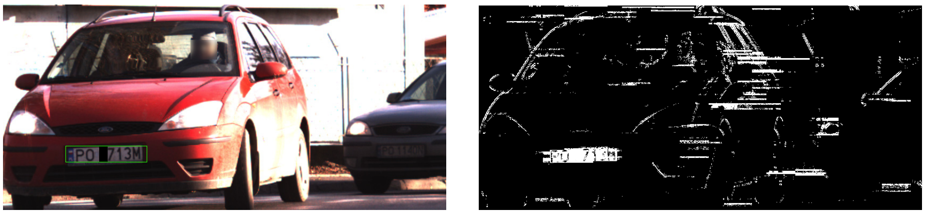

Fig. 1. The output (right) of the best-of-run individual of one of the GP-IFS runs, when applied to the input image shown in left inset. Pixel brightness reflects individual's confidence in the presence of plate. The plate has been obfuscated due to privacy concerns.

recognizer together. In general, a plate reader cannot be guaranteed to correctly segment every plate and correctly recognize all segmented characters. In all recognition systems considered in this experiment, we use the same character segmentation algorithm and the same recognizer (an SVM classifier trained on a separate data set of character images). It is then justified to ask: what is the maximal fitness that can be attained by the entire system equipped with this plate reader, given a *perfect* plate detector?

We answer that question by applying the plate reader to actual plate locations in images, manually determined by a human expert. It turns out that for the training set, such a system does not work perfectly, attaining $f_{std} = 0.9196$. The data presented in Table 2 shows that GP-IFS is very close to this limit. As a matter of fact, two out of 10 runs of GP-IFS reach this performance. For the testing set, the system based on human labeling reaches $f_{std} = 0.8694$. This confrontation proves that the performance of recognition systems produced by both GP and GP-IFS is much closer to the realistic upper limit that it may appear when judging from the fitness definition alone.

Figure 1 presents exemplary results of the plate detection process carried out by the best-of-run individual of one of the IFS runs. Light-colored regions indicate the locations where the filter's belief in plate presence is higher. It may be observed that the detector correctly identifies the location of the plate, and does not get distracted by regions that have color characteristics similar to license plates (e.g., the light background behind the car).

Table 3 presents the distributions of Levenshtein distance for the training and test set for the GP-IFS approach. Most errors consist in single-character mistakes, which suggests that once the plate is correctly detected, the plate can be read flawlessly or with a low number of mistakes. For the testing set, the share of perfect recognitions drops on average to 55.7 percent, from 82.1 for the training set, but relying on the best of best-of-run individuals (last row of the table) leads to substantial improvements.

The median size of evolved image programs, measured in the number of tree nodes (leaves and nonterminal tree nodes) is 136. However, one of the very well performing best-of-run systems (training fitness 0.9175, testing fitness 0.7122)

Table 3. Distribution of Levenshtein distance d for GP-IFS (percents)

Levenshtein distance d	0	1	2	3	4	5	> 5
Average on training set	82.1	10.2	1.3	0.0	2.1	3.1	1.2
Average on testing set	55.7	11.9	3.2	2.6	1.0	2.1	23.5
Best-on-training on testing set	59.8	14.4	6.2	4.1	1.0	0.0	14.4

comprised mere 76 nodes. Though this number is still quite weighty, we hypothesize that most of the evolved systems can be substantially simplified without significant impact on fitness. This however, requires a separate investigation.

Analysis of IFS Weights. To verify the impact of example weighing realized by IFS, we collected additional statistics. In each generation, for each training image t, we logged $\sum_{s' \in P} f(s', t)$, i.e., the sum of individuals' performances on t. This quantity, called *solvability* in following, occurs in the denominator of Eq. (3), and its reciprocal determines the weight of an example. Thus, the easier an example appears to the individuals in the current population, the greater its solvability. In Fig. 2, we plot the average, minimum, and maximum of solvability, calculated over all 200 generations of an exemplary run, individually for each training example. The examples have been sorted ascendingly with respect to average solvability. As there are 100 individuals in population and the performance on a single example is normalized to $[0, 1]$ interval (Eq. 4), solvability cannot exceed 100.

Analysis of Fig. 2 allows us to draw several conclusions. It turns out that our training set contains eight examples, grouping on the left, that have never been correctly recognized by *any* individual throughout the run. Apparently, these examples turned out to be too difficult for the capabilities of learners. It is interesting to notice that they could have been safely discarded from the training set, as, by remaining constantly unsolved, they never contributed to fitness differences between individuals, so they had virtually no impact on the learning process.

Looking at the right-hand part of the graph, there are no examples that have been solved by all individuals in all generations, although there are a dozen or two of them for which this was true in at least one generation (see the *max* curve). Roughly speaking, of all 97 examples, around 60 rightmost can be judged as easy, with the average solvability of 87.0 or more. However, contrary to the left part of the graph, these examples definitely contributed to the search process, as in some generations they have been solved by no more than 20 individuals (see the *min* curve).

Finally, Fig. 2 suggests that the most valuable part of our training set comprises roughly 30 examples, with indices from 9 to about 38. These frames were moderately challenging to the learners, and probably contributed the most to fitness variation. Interestingly, these examples may be considered as an analog to the concept of *ideal evaluation set* coined in coevolutionary algorithms [4]. It may be hypothesized that recognition systems trained only on these examples could attain decent performance.

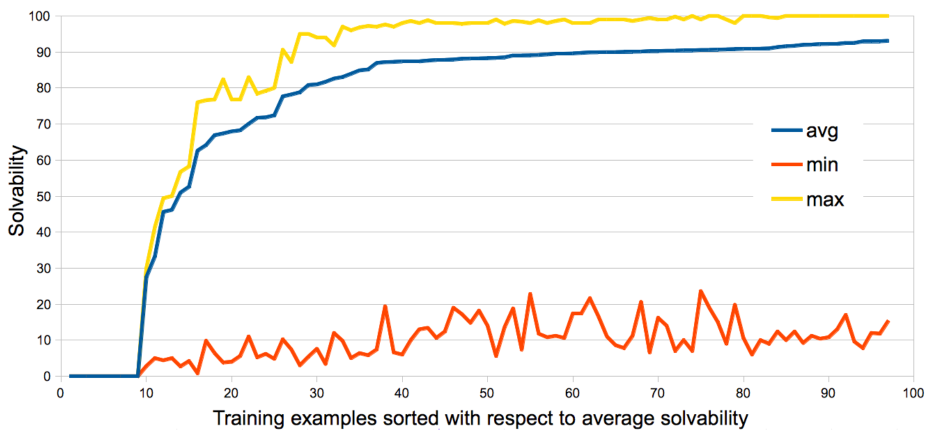

Fig. 2. Distribution of average, minimum, and maximum solvability of training examples, as estimated by IFS

6 Conclusion

The main conclusion of this study is that implicit fitness sharing is a useful alternative to standard evaluation when solving an object detection task. Because IFS is a general technique that abstracts from the nature of examples, it is justified to claim that some gains can be attained when evolving programs for other types of visual tasks, like image processing or object classification.

The recognition accuracy reported here is substantially greater than the one we obtained when using evolutionary algorithm to only *tune* the parameters of a fixed plate detection system [8]. On one hand, this result was expected, as the space of all possible image analysis programs is much greater than the space of parameters of a fixed image analysis program, and better (or at least not worse) performing programs can be found in such a bigger space. On the other hand, more expressive representation of solutions (complete programs vs. vectors of parameters) makes overfitting more likely, so running the above experiments was necessary to verify the test-set performance of plate recognition systems.

Typically, a passing-by car is captured in a few consecutive frames. By applying the approach reported above to multiple frames and aggregating the results, additional boost in recognition accuracy can be obtained. This is one of extensions that we intend to investigate in the follow-up of this study.

Acknowledgment. M. Nawrocki acknowledges support from European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°218086. K. Krawiec acknowledges support from National Science Center grant DEC-2011/01/B/ST6/07318.

References

1. Abdullah, S., Khalid, M., Yusof, R., Omar, K.: License plate recognition using multi-cluster and multilayer neural networks 1, 1818–1823 (2006)
2. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
3. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), software, `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
4. de Jong, E.D., Pollack, J.B.: Ideal Evaluation from Coevolution. Evolutionary Computation 12(2), 159–192 (2004)
5. For, W.K., Leman, K., Eng, H.L., Chew, B.F., Wan, K.W.: A multi-camera collaboration framework for real-time vehicle detection and license plate recognition on highways, 192–197 (June 2008)
6. Ji-yin, Z., Rui-rui, Z., Min, L., Yin, L.: License plate recognition based on genetic algorithm. In: 2008 International Conference on Computer Science and Software Engineering, vol. 1, pp. 965–968 (2008)
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
8. Krawiec, K., Nawrocki, M.: Evolutionary Tuning of Compound Image Analysis Systems for Effective License Plate Recognition. In: Jędrzejowicz, P., Nguyen, N.T., Hoang, K. (eds.) ICCCI 2011, Part I. LNCS, vol. 6922, pp. 203–212. Springer, Heidelberg (2011)
9. Luke, S.: ECJ evolutionary computation system (2002), `http://cs.gmu.edu/~eclab/projects/ecj/`
10. Martinsky, O.: Algorithmic and mathematical principles of automatic number plate recognition systems (2007)
11. McKay, R.I.B.: Committee learning of partial functions in fitness-shared genetic programming. In: 26th Annual Conference of the IEEE Third Asia-Pacific Conference on Simulated Evolution and Learning 2000, Industrial Electronics Society, IECON 2000, October 22-28, vol. 4, pp. 2861–2866. IEEE Press, Nagoya (2000), `http://sc.snu.ac.kr/PAPERS/committee.pdf`
12. McKay, R.I.B.: Fitness sharing in genetic programming. In: Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., Beyer, H.G. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000), July 10-12, pp. 435–442. Morgan Kaufmann, Las Vegas (2000), `http://www.cs.bham.ac.uk/~wbl/biblio/gecco2000/GP256.pdf`
13. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods – Support Vector Learning, MIT Press, Cambridge (1998)
14. Smith, R., Forrest, S., Perelson, A.: Searching for diverse, cooperative populations with genetic algorithms. Evolutionary Computation 1(2) (1993)
15. Sun, G., Zhang, C., Zou, W., Yu, G.: A new recognition method of vehicle license plate based on genetic neural network. In: 2010 the 5th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 1662–1666 (2010)

16. Tseng, P.C., Shiung, J.K., Huang, C.T., Guo, S.M., Hwang, W.S.: Adaptive car plate recognition in qos-aware security network. In: SSIRI 2008: Proceedings of the 2008 Second International Conference on Secure System Integration and Reliability Improvement, pp. 120–127. IEEE Computer Society, Washington, DC (2008)
17. Wang, F., Zhang, D., Man, L.: Comparison of immune and genetic algorithms for parameter optimization of plate color recognition. In: 2010 IEEE International Conference on Progress in Informatics and Computing (PIC), vol. 1, pp. 94–98 (2010)
18. Zimic, N., Ficzko, J., Mraz, M., Virant, J.: The fuzzy logic approach to the car number plate locating problem. In: IASTED International Conference on Intelligent Information Systems, p. 227 (1997)