

Genetic Programming for Mathematical Morphology Algorithm Design on Binary Images

Marcos I. Quintana

School of Computer Science
University of Birmingham
Birmingham, B15 2TT
United Kingdom

M.I.Quintana@cs.bham.ac.uk

Riccardo Poli

Department of Computer Science
University of Essex
Colchester, CO4 3SQ
United Kingdom

RPoli@essex.ac.uk

Ela Claridge

School of Computer Science
University of Birmingham
Birmingham, B15 2TT
United Kingdom

E.Claridge@cs.bham.ac.uk

Abstract

This paper presents a Genetic Programming (GP) approach to obtain Mathematical Morphology (MM) algorithms for binary images. The algorithms are constructed using the basic MM operators, i. e. erosion and dilation. However differently to previous approaches, we explore the utilisation of irregular structuring elements of various sizes. GP is used to find (not fixed-length) procedures to convert a binary image into another containing just a particular characteristic of interest. The analysis of results includes two similarity fitness functions, training sets with different numbers of elements and different sizes of the training images over three different objectives.

1 Introduction

MM is well known as a powerful tool for various image processing tasks [Serra, 1982]. It is suitable for shape related processing since morphological operations are directly related to the object shape. To design a MM procedure (i. e. algorithm) some expert knowledge is necessary to properly select the structuring elements and to make an adequate selection of the morphological operators sequence. It has been suggested that Genetic Algorithms (GAs) [Holland, 1975] could have good performance for morphological filter design [Harvey and Marshall, 1996]. Furthermore, GAs have obtained promising results in the automatic acquisition of MM procedures [Yoda *et al.*, 1999]. Previous research has been limited to optimising one filter (for gray-level images) [Yu *et al.*, 1998] or to obtain fixed-length sequences of morphological operators using a reduced number of structuring elements [Bala and Wechsler, 1991]. GP is the extension of GAs in which the structures that make up the population under optimisation are not fixed-length strings that encode possible solutions to a problem, but *programs* that, when executed, *are* the candidate solutions to the problem [Koza, 1992]. In this paper GP is used to search for solutions to particular tasks in the MM algorithm's search space. This has the potential to outperform previous approaches because it explores a bigger space (not fixed-length solutions) and it includes both regular and irregular structuring elements of various sizes. In section 2

a brief introduction to MM operators is presented. In section 3 we review how GAs have been used for morphological image analysis. In section 4 GP is introduced as an optimisation tool. We describe the GP method proposed for morphological algorithm design in section 5. The results are presented and analysed in section 6. Finally some conclusions and suggestions of further research are presented in section 7.

2 Mathematical Morphology

MM [Serra, 1982] is a relatively separate part of image analysis. The non-morphological approach to image processing is close to calculus, being largely based on the point spread function concept (e. g. Dirac impulse) and linear transformations such as convolution. MM is based on geometry and shape; morphological operations simplify images, and preserve the main shape characteristics of objects. MM is basically a tool for extracting image components that are useful in the representation and description of region shape, such as boundaries and skeletons. Morphological techniques are also used for pre- or post-processing, such as morphological filtering, thinning and pruning.

The language of MM is *set theory*. As such, morphology offers a unified and powerful approach to numerous image processing problems. Sets in MM represent the sets of objects in an image. For example, the set of all black pixels in a binary image is a complete description of the image. In binary images, the sets in question are members of the 2-D integer space Z^2 , where each element of a set is a tuple whose coordinates are (x,y) of a black pixel in the image. Gray-scale digital images can be represented as sets whose components are in Z^3 . Sets in higher dimensional spaces can contain other image attributes, such as color and time varying components.

Morphological operations are predominantly used for image pre-processing (noise filtering, shape simplification), enhancing object structure (skeletonizing, thinning, thickening, convex hull, object making) and quantitative description of objects (area, perimeter, projections, Euler-Poincare characteristics) [Gonzalez and Woods, 1992].

2.1 Basic Morphological Operations

The two morphological operations most used in MM are *dilation* and *erosion* and most of the algorithms developed by experts to perform a particular task make use of them.

With A and B as sets in Z^2 and \emptyset denoting the empty set, the dilation of A by B , denoted $A \oplus B$, consists of obtaining the reflection of B about its origin, then shifting this reflection \hat{B} by x to obtain $(\hat{B})_x$. The dilation of A by B , as shown in Equation 1 is the set of all x displacements such that $(\hat{B})_x$ and A overlap by at least one nonzero element. Figure 1 exemplifies this operation.

$$A \oplus B = \{x | (\hat{B})_x \cap A \neq \emptyset\} \quad (1)$$

$$A \ominus B = \{x | (\hat{B})_x \subseteq A\} \quad (2)$$

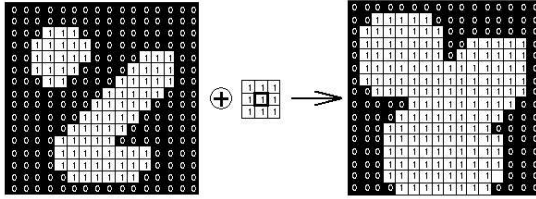


Figure 1: Dilation.

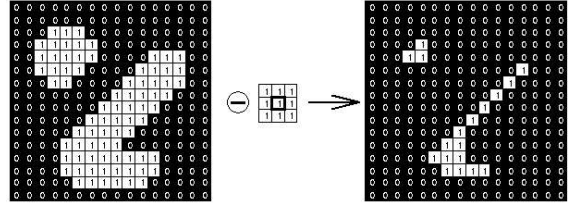


Figure 2: Erosion.

For sets A and B in Z^2 the erosion of A by B , denoted $A \ominus B$, is defined as the set of all points x such that \hat{B} translated by x is contained in A . The erosion of A by B , as shown in Equation 2 is exemplified in Figure 2. These two equations are not the only definitions for dilation and erosion, but they are usually preferred in practical implementations because of their analogy with the operation of convolution for linear filtering.

3 GAs for MM filters and procedure design

Morphological filters are an important class of non-linear digital signal processing and analysis filters, which have found a range of applications, giving excellent results in areas such as noise reduction, edge detection and object recognition. However, few design methods exist for these morphological filters, with selection of filter type and sequence of application tending to be both ad-hoc and application specific. The number of possible structuring elements (also known as kernels) from which to choose is very large and this, combined with the possible choices of morphological operator sequences, indicates the complexity of the morphological filter design.

In the case of morphological filters there are two basic parameters: the structuring elements and the sequence of morphological operations. It has been demonstrated how simple GAs can be employed in the search for morphological filters realising optimum performance in a given image processing task [Harvey and Marshall, 1996]. In that approach it was used a fixed number of morphological operators including a do-nothing operator to avoid the problem of using only fixed sized sequences. It seems that this do-nothing operator appear very often in the chromosomes, indicating how important is to let the search use non fixed-length solutions.

It has been suggested to obtain MM procedures showing an original binary image and a hand-made goal image. The search space of morphological sequences is explored using GAs [Yoda *et al.*, 1999]. The solution representation uses 4-20 fixed-length chromosomes with four types of regular structuring elements. After 4 examples of different complexity, it is clear that the proposed method is fairly good to obtain a near optimum solution, but the required computational effort increases when the task to perform is more complex. Figure 3 exemplifies how it is expected to get a goal image from the original after a particular transformation procedure.

In section 5 we show how GP can be used to overcome the problems presented. Before we provide more details on this, we briefly describe how GP is used to evolve programs in section 4.

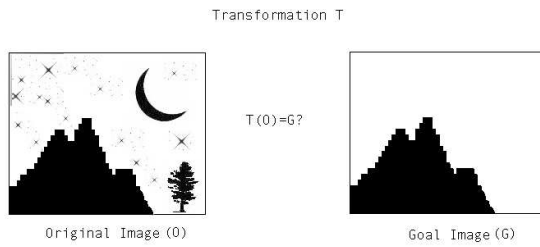


Figure 3: Transformation from Original Image (O) to Goal Image (G).

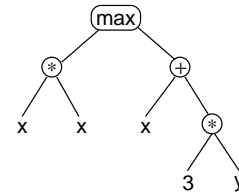


Figure 4: Syntax-tree representation of the expression $\max(x*x, x+3*y)$.

4 Genetic Programming

GP is an extension of GAs where the chromosomes represent *computer programs*. Programs are expressed in GP as syntax trees, rather than as lines of code. For example, the simple expression $\max(x*x, x+3*y)$ would be represented as shown in Figure 4.

The set of possible internal (non-leaf) nodes used in GP syntax trees is called the *function set*, $\mathcal{F} = \{f_1, \dots, f_{N_F}\}$. The set of terminal (leaf) nodes in the syntax trees representing programs in GP is called the *terminal set* $\mathcal{T} = \{t_1, \dots, t_{N_T}\}$.

The *search space* of GP is the set of all the possible (recursive) compositions of the functions in $\mathcal{F} \cup \mathcal{T}$. The basic search algorithm used in GP is a classical GA with mutation and crossover specifically designed to handle syntax trees. GP starts with an initial population of randomly generated computer programs composed of functions and terminals appropriate to the problem domain. The algorithm is controlled by a fitness function which evaluates the quality of the programs stochastically produced by GP. This typically requires running such programs (within the GP environment) on a number of test cases. The best individuals are selected and modified to produce a new population which is evaluated in the next generation. This process is repeated until a stop criterion is reached.

Unlike non-learning techniques GP solves specific-domain problems without being explicitly programmed. Because it directly searches the space of computer programs, its outputs (computer programs) can be immediately interpreted and used. Thanks to these two properties GP is an excellent tool to provide optimum or near optimum algorithms to solve problems which are difficult to understand and solve for humans, like some problems of Image Analysis.

GP has been applied successfully to solve some difficult problems in Image Analysis like classification [Teller and Veloso, 1995], filter evolution for different purposes [Poli, 1996] and evolved agents for images. To the best of our knowledge, no use of GP for morphological image analysis has been reported in the literature. In section 5 we describe our first experiments on GP for MM on binary images.

5 GP for MM algorithm design

5.1 Process

The GP approach suggested assumes that it is possible to find a sequence of morphological operators in the MM algorithm's search space to convert an image into another containing only a particular feature of interest. To show this idea we select musical sheets as examples and extract some features from them.

The process is visualised in Figure 5. The first step is to create some examples of correct feature extraction by hand to be used as training sets in GP. Next it is necessary to define the type, size and number of structuring elements to be used in the GP search. After the *primitive set* is defined we start the GP search to obtain a (near) optimum tree representing a MM algorithm sequence (see subsection 5.5). Note that it won't be a fixed-length sequence. The best evolved sequence, according to the fitness value assigned for the fitness function, is also analysed visually to decide whether or not the numerical fitness value is reflected in the image quality. The features used in this process are described in the next subsections.

5.2 Structuring Elements

When a MM algorithm is designed by hand, the programmer usually chooses a regular structuring element to make a sequence of operations, there is no reason for choosing a regular element except that we as humans, are more likely to understand *regularities* such as squares, lines, triangles, etc. It is interesting to note that the MM algorithm's search space is not limited to *regular* structuring elements, but that there are many *irregular* structuring elements usually ignored when designing a MM algorithm. We suggest to include irregular structuring elements (selected randomly) in the GP search space. We use structuring elements of sizes 3×3 , 5×5 and 7×7 such as those shown in Figure 6. In our search we include 11 regular and 11 irregular structuring elements of each size.

5.3 Training Sets

We test GP to search for MM algorithms looking for procedures to extract three different features: heads, hooks and lines. To find out whether or not the size of the images in the training sets affects the results quality, we made by hand training sets containing images of size 16×16 , 32×32 and 64×64 for each one of the different features to look for. A set of images belonging to the 64×64 training set is shown in Figure 7. To learn if the number of elements in the training set affects the results quality we use four different numbers of elements in the training set (1,5,15 and 25).

5.4 Fitness Functions

All our experiments were performed using two different fitness functions to evaluate the similarity between two images. The objective fitness function A is known as *similarity* ($0 \leq A \leq 1$) [Yoda *et al.*, 1999], a normalized correlation coefficient between a goal and a processed image defined as

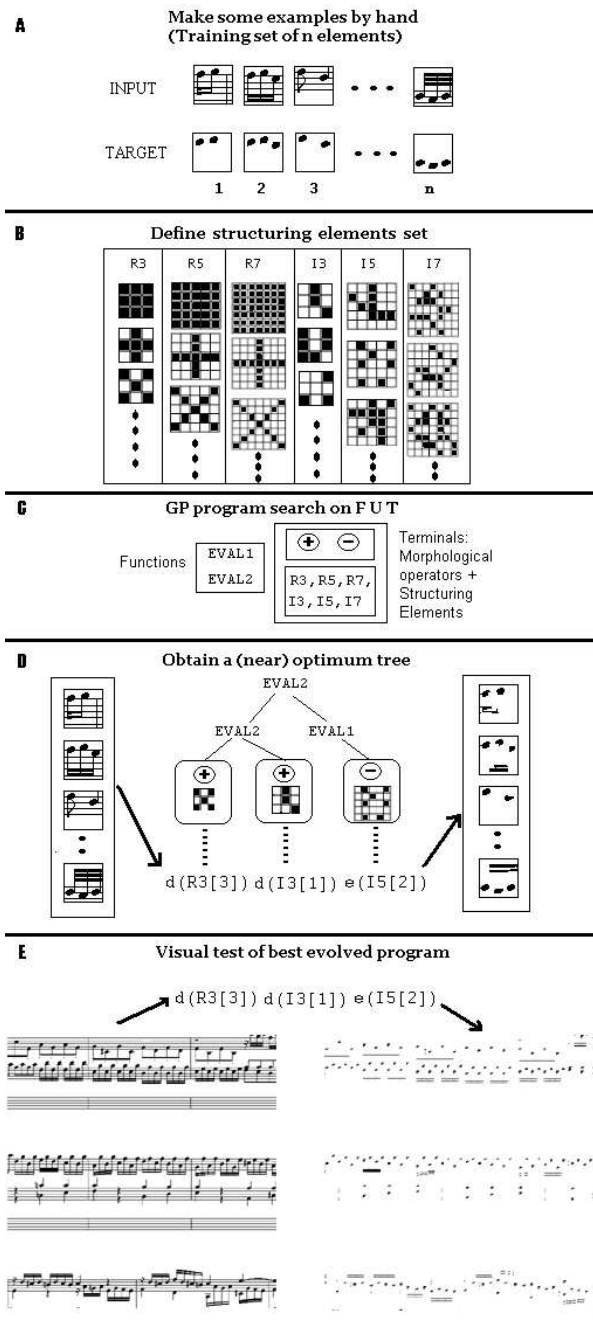


Figure 5: Process to obtain a MM algorithm using GP. A) Make examples by hand. B) Decide structuring elements to use. C) Perform GP search. D) Obtain a (near) optimum tree. E) Evaluate best result.

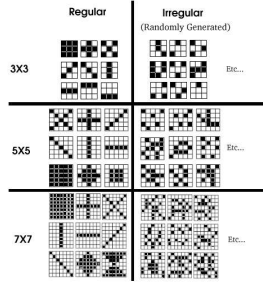


Figure 6: Examples of various sizes regular and irregular structuring elements.

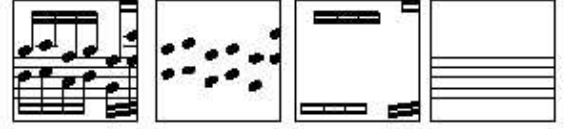


Figure 7: One example of the 64×64 training set. Fragment of music sheet. Handmade: heads, hooks and lines.

$$A = \frac{(f \cdot g)}{\sqrt{(f \cdot f)}\sqrt{(g \cdot g)}}; \text{ where } (f \cdot g) = \frac{1}{N} \cdot \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^N f(i, j) \cdot g(i, j) \quad (3)$$

The objective fitness function B ($0 \leq B \leq 1$) is related to the *trade-off* between *sensitivity* (SV) and *specificity* (SP) needed on detection algorithms.

$$B = 1 - \frac{\sqrt{(1 - SP)^2 + (1 - SV)^2}}{\sqrt{2}}; \text{ where } SV = \frac{TP}{TP + FN}; \text{ and } SP = \frac{TN}{FP + TN} \quad (4)$$

TP is the number of *true positives*, FP is the number of *false positives*, TN is the number of *true negatives* and FN is the number of *false negatives*. For these experiments the goal is to convert the original image (O) into the goal image (G). If the pixel $G(x_i, y_i) = 1$ then is a true positive if $O(x_i, y_i) = 1$ and a false positive if $O(x_i, y_i) = 0$. If the pixel $G(x_i, y_i) = 0$ then is a true negative if $O(x_i, y_i) = 0$ and a false negative if $O(x_i, y_i) = 1$.

5.5 GP features

We use a function set including two functions namely EVAL1 and EVAL2 of arity 1 and 2 respectively. The terminal set includes embedded strings $x(yz[w])$ where $x \in \{e, d\}$, represents the morphological operators (erosion and dilation); $y \in \{R, I\}$, represents the type of structuring element selected (regular and irregular); $z \in \{3, 5, 7\}$, represents the size of structuring element and $w \in \{1..11\}$ represents the structuring element index. In Figure 5.D is exemplified how the obtained trees are transformed to linear representation. The obtained strings are straightforward used as MM algorithms (read left to right) to be applied over the training set during the evolution process. The GP features are those suggested for a simple GP in the literature [Koza, 1992]. Population of 50 programs over 100 generations (to limit the computational effort). Maximum tree depth of 10. A 0.9 crossover rate

and 0.1 mutation rate using a *half and half* initialisation method. It is important to remark that the aim of these experiments is to look for interesting patterns in the solutions rather than performing a thorough comparison of performance.

6 Results and Analysis

We made a total of 1512 GP runs combining 3 features to extract (heads, hooks and lines), 4 different numbers of training set elements (1, 5, 15, 25), 2 different fitness functions, 3 different training set image sizes (16×16 , 32×32 and 64×64), 3 different combinations of structuring element types (R, I and RI) and 7 different combinations of structuring elements sizes (3, 5, 7, 35, 37, 57 and 357). Although the visual analysis made was not exhaustive (i. e. we focus on those MM algorithms with a good fitness value or with interesting string features) we made some interesting observations.

The fitness function B is more accurate than the fitness function A. The second has tendency to maximise either *sensitivity* or *specificity*. In our approach we avoid using human expertise according to the problem at hand (i. e. the three different MM feature extraction algorithms were obtained using GP in the same way). A possibility could be to use the fitness function described in [Poli, 1996], but note that in such a case expert knowledge will be required for fitness function tuning.

GP easily found different algorithms to solve the *heads* extraction problem. The degree of accuracy is very difficult to evaluate visually and, depending on the evaluator it does not always match the numerical fitness values obtained. However, we believe that many of the results obtained are as good as those that could be obtained by an expert writing the algorithm by hand. Figure 8 shows an image for visual inspection and Figure 9 shows the result of an MM algorithm generated by GP. The *hooks* extraction problem is a fairly difficult task. That feature could be mismatched with lines, heads or other features present in a musical sheet. In spite of that, some GP generated algorithms present good approximations to the desired task. One example is presented in Figure 11. Contrary to expectation, the *lines* extraction problem was the one presenting the most difficulties for the GP approach proposed. The results obtained were either tending to completely white images or to mismatch the lines with other features. We show an example in Figure 10 where GP accurately finds the lines, but also includes most of the hooks present in the test image.

The results are improved when combining structuring elements of different sizes and types. The irregular kernels are very useful as complement but are poor in performance when used by themselves. It should be noted that small structuring elements obtain better performance when used alone. It should be preferred to use a small number (i. e. the 1-5 training sets obtained better results than the 15-25). The size of the images used in the training set didn't affect the final results much, except perhaps 32×32 case was slightly better.

7 Conclusions

In this paper we have presented an approach to morphological image analysis based on the idea of using GP to design MM algorithms for binary images. We have analysed the behavior of two different fitness functions over various examples on musical sheets using 3×3 , 5×5 and 7×7 sized regular



Figure 8: Example of testing image for visual purposes.

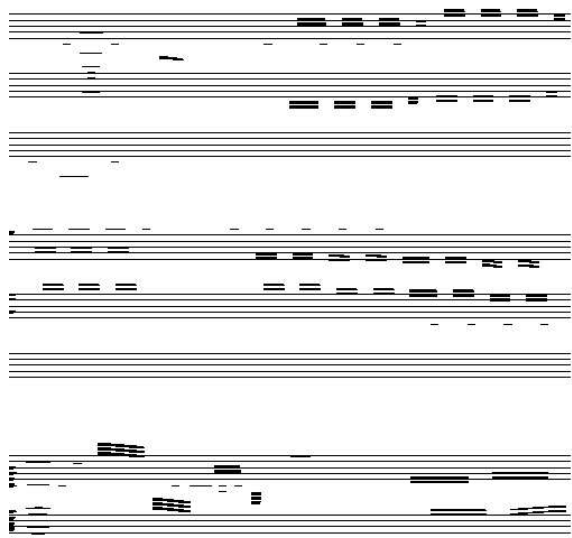


Figure 10: Example of a good visual result for lines on the image from Figure 8.

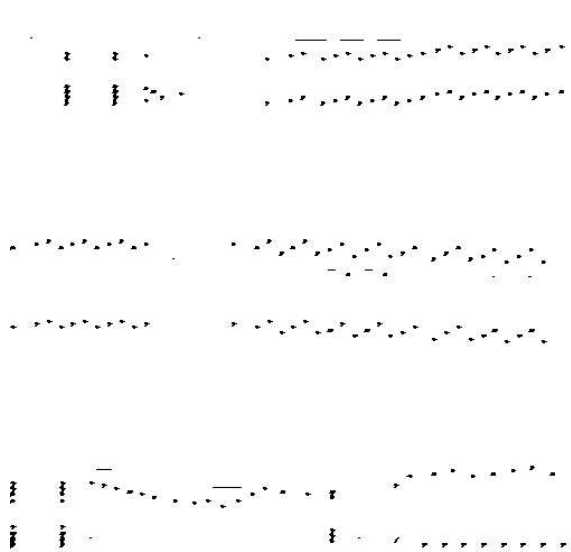


Figure 9: Example of a good visual result for heads on the image from Figure 8.

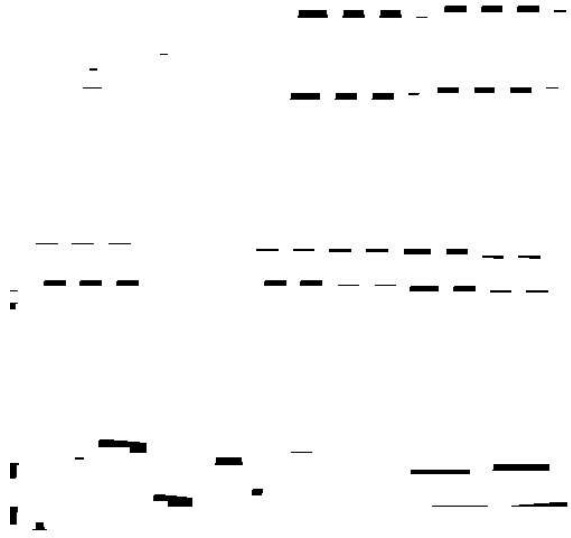


Figure 11: Example of a good visual result for hooks on the image from Figure 8.

and irregular structuring elements. We have also analysed how the size of the training set of images and the number of them may affect the results quality. We have concluded that fitness function A has a tendency to maximise either *sensitivity* or *specificity*. Fitness function B is more accurate but it is still not good enough for optimal results. The performance is improved when combining structuring elements of different sizes and types.

Acknowledgments

The work presented in this paper was funded by the School of Computer Science at the University of Birmingham and Conacyt (Mexico). Many thanks to Tim Kovacs, Maria Barilla, Hector Montes, Raymundo Marcial and Felipe Orihuela for useful comments on this paper.

References

- [Bala and Wechsler, 1991] J. Bala and H. Wechsler. Shape analysis using morphological processing and genetic algorithms. In *Proceedings of the 1991 IEEE International Conference on Tools with Artificial Intelligence TAI'91*, pages 130–137, Los Alamitos, CA., November 1991. IEEE Computer Society Press.
- [Gonzalez and Woods, 1992] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley, Reading, MA, USA, 3rd edition, 1992.
- [Harvey and Marshall, 1996] N.R. Harvey and S. Marshall. The use of genetic algorithms in morphological filter design. *Signal Processing: Image Communication*, 8(1):55–72, January 1996.
- [Holland, 1975] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, USA, 1975.
- [Koza, 1992] John R. Koza. *Genetic programming: On the programming of computers by natural selection*. MIT Press, Cambridge, Mass., 1992.
- [Poli, 1996] Riccardo Poli. Genetic programming for image analysis. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 363–368, Stanford University, CA, USA, 28–31 July 1996. MIT Press.
- [Serra, 1982] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.
- [Teller and Veloso, 1995] Astro Teller and Manuela Veloso. Algorithm evolution for face recognition: What makes a picture difficult. In *International Conference on Evolutionary Computation*, pages 608–613, Perth, Australia, 1–3 December 1995. IEEE Press.
- [Yoda *et al.*, 1999] I. Yoda, K. Yamamoto, and H. Yamada. Automatic acquisition of hierarchical mathematical morphology procedures by genetic algorithms. *Image and Vision Computing*, 17(10):749–760, 1999.
- [Yu *et al.*, 1998] M. Yu, N. Eua-anant, A. Saudagar, and L. Udpa. Genetic algorithm approach to image segmentation using morphological operations. In *International Conference on Image Processing*, volume 3, pages 775–779, 1998.