Hand posture recognition using real-time artificial evolution

Benoit Kaufmann[1], Jean Louchet[2], and Evelyne Lutton[1]

[1] INRIA Saclay, Parc Orsay Université, 4 rue Jacques Monod, 91893 Orsay Cedex
benoit.kaufmann@gmail.com, evelyne.lutton@inria.fr
[2] ARTENIA, 24 rue Gay Lussac, 92320 Chatillon
jean.louchet@gmail.com

Abstract. In this paper, we present a hand posture recognition system (configuration and position) we designed as part of a gestural man-machine interface.

After a simple image preprocessing, the parameter space (corresponding to the configuration and spatial position of the user's hand) is directly explored using a population of points evolved via an Evolution Strategy. Giving the priority to exploring the parameter space rather than the image, is an alternative to the classical generalisation of the Hough Transform and allows to meet the real-time constraints of the project. The application is an Augmented Reality prototype for a long term exhibition at the Cité des Sciences, Paris. As it will be open to the general public, rather than using conventional peripherals like a mouse or a joystick, a more natural interface has been chosen, using a microcamera embedded into virtual reality goggles in order to exploit the images of the user's hand as input data and enable the user to manipulate virtual objects without any specific training.

1 Introduction

The work described in this paper is part of the REVES project[3], whose aim is to create a prototype for a permanent exhibition at the *Cité des Sciences et de l'Industrie de Paris*[4], called *«Objectifs Terre : La Révolution des satellites»*[5] (objective: Earth). One of the components of this exhibition is devoted to artificial satellites and includes a virtual reality device that enables the user to interact with them.

The user's interface is based on the recognition of hand postures, using a fast generalised Hough transform operated using artificial evolution. The augmented reality device is described in Section 1.1 and the gestural interface in Section 1.2. We present the evolutionary Hough transform methology in Section 2, then describe two different implementations of hand model determination in Sections 3 et 4. Results are presented in Section 5 and the conclusions in Section 6.

[3] REVES ANR (French National Research Agency) contract No 2160 (2007-2009)
[4] http://www.cite-sciences.fr/english
[5] http://www.cite-sciences.fr/objectifs-terre

1.1 The Augmented Reality device

The public will be staying around a large table. Above the table is a videoglobe [6] which displays an animated image of the Earth. Each user is provided with augmented reality "see through" goggles[7] (see figure 1) which allow to display the satellites that turn over the globe, or two- or three-dimensional information when the user is looking down at the table. To this end, the goggles are fitted with a miniature camera. Processing the images from this camera allows to position the synthetic images (e.g. the satellite images) displayed by the goggles relative to what the user can directly see through the goggles. It also allows the user to manipulate the objects displayed, by analysing the user's hand gestures.

Our contribution to this project was to design the algorithms that allow real-time recogniton of hand posture and position from the embedded camera and enable the user to interact with the virtual objects.

The googles. Example of usage.

Fig. 1. The interactive device (courtesy Zile Liu, Laster Technologies).

1.2 The gestural interface

In order to ensure natural interaction and allow the user to move whithin the available multimedia content, the interface should provide at least an object designation tool. The object may be e.g. a 3D object, an hypertext link, an icon or a menu entry. Selecting it allows to visualise richer information or move into different menu options. As this is assumed to replace the usual mouse click, we had to devise a simple communication language based on hand configuration.

In addition to this, it is often useful to create a pointer which indicates where the system did locate the user's hand, in order to correct the eye-to-camera parallax and reduce the uncertainty that goes with the size of the detected patterns (e.g. pointed index or the whole hand). Then it is necessary to define two distinct gestures: one to move the pointer, and one to activate the object that has been pointed. The object is selected by changing the hand configuration. The advantage of the latter is the possibility to implement extra primitives such as rollover which allow access to other pieces

[6] Developed by Agenium (http://www.agenium.eu/sections.php?op=listarticles&secid=40)
[7] Developed by Laster Technologies (http://www.laster.fr/?lg=en)

of information by changing the pointer shape or the object colour. It is also possible to implement commands that move the display zone, or rotate a 3-D object.

This could have been made possible through using the mouse buttons: a long pressure would switch from one shape to another one, and releasing the button would get back to the first shape. In fact we implemented this functionality by interpreting changes in the user's hand posture: closing and opening the hand is given the same semantics as pressing and releasing a mouse button. This improves the user's feeling to actually interact physically and grasp the object in order to turn or move it. Other display parameters (e.g. the scale) may be edited through creating cursors and moving them a similar way. It is also possible to relate the scale change to the apparent scale of the hand: a grasp plus a movement toward the user corresponds to zooming in, and reversely for zooming out.

The major problem to be solved here is the robust and real time detection of several hand gestures (at least, an open and a closed hand), without prior calibration or learning. To this end we developed an evolutionary Hough transform.

There exists an abundant literature about hand detection, but most of the proposed methods were not adapted to our conditions of use: for instance some methods are based on the detection of the entire body[1–3], or at least the user's foreharm[4] to deduce the position of the hands. This solution does not fit with our device because the camera is located on the user's face and therefore can only see the hand and not the rest of the body. Other methods only detect the hand but need a uniform[5] or at least a fixed background[6]. Because the camera we are using is head-mounted, the user may turn their head, which makes the background change and show moving objects that could be wrongly detected as hands and disturb detection. For the same reason, motion-based[7] and multi-camera[8, 9] detection methods cannot be applied.

2 EvHough, evolutionary exploration of a parameter space

The Hough Transform[10] and its classical generalisations[11] fundamentally scan the image looking for a certain type of local features. Each time such a feature has been found, it writes into the parameter space by reversing a model in order to increment the number of votes for the parameter values that are able to provide an explanation to the feature found. As the model is usually not injective, if n is the dimension of the set of parameters, each detected feature generates into the parameter space a variety with dimension $n-1$: a curve if $n = 2$, a surface if $n = 3$, etc. This results into a very slow process with $n = 3$ and unrealistic processing times beyond.

The alternative proposed in [12] then in [13] consists in a direct, smart exploration of the parameter space, using a heuristics given by the Artificial Evolution paradigm. It does not involve any domain-specific knowledge (here, knowledge in image processing) except what has been expressed through the so-called "synthesis model" that allows to calculate the features corresponding to any point in the parameter space.

To this end, the algorithm creates a randomly initialised population of points in the parameter space, then evolves it using genetic operators: selection, mutation, crossover [14–16]. The selection process is based on a fitness function which, for each point in

the parameter space, executes the model and calculates a value of similarity between the synthesised and the actual feature.

Thus, in the simplest case of sraight line detection, for each pair (ρ, θ), the evolutionary Hough algorithm will calculate the equation of the corresponding line, then evaluate e.g. the number of interest points on this line or the average contrast along it, in order to get the fitness value of the parameter pair (ρ, θ) in question.

The main interest of this is to open the possibility to work efficiently with high dimensional parameter spaces, where a direct Hough-style approach would fail. The main limitation is a classical one in the field of evolutionary algorithms: it remains uneasy to decide when the algorithm has to be terminated. However, in practice this is not a real issue: artificial evolution is naturally time-compliant and, unlike most classical image processing algorithms, may work on data that can be changed during processing. As shown below, this can be an interesting point when processing image sequences in real time.

3 Detection of hand models

3.1 Preliminary processing

The searched models are hand contours shapes, encoded as lists of points, in order to keep the number of points to be tested low. We thus apply a series of filters to the video sequences to be analysed.

Four convolution filters are first used to compute the gradient in 4 directions (vertical, horizontal and two diagonals). For each pixel of the input image, the largest of these 4 values then yields the gradient direction and value. The same information is also kept in memory for each point of the models.

In order to accelerate the computation and facilitate the optimisation process, a proximity function (equation 1) is computed at each frame, and gives for each x of the input image a measurement of its proximity to a contour[17].

$$prox(x) = max_{y \in image}(grad_y / dist_{x,y}) \tag{1}$$

where

- x and y are two pixels of the input image,
- $grad_y$ is the gradient value of pixel y, as defined earlier,
- $dist_{x,y}$ is the distance between x and y according to equation 2[8], with x_i, for $i = 1..8$, the 8 immediate neighbours of x, as defined on figure 2.

$$\begin{aligned} dist_{x,x} &= 1 \\ if \quad x \neq y, \quad dist_{x,y} &= min(dist_{x_1,y} + \sqrt{2}, dist_{x_2,y} + 1, dist_{x_3,y} + \sqrt{2}, \\ & dist_{x_4,y} + 1, dist_{x_5,y} + 1, dist_{x_6,y} + \sqrt{2}, dist_{x_7,y} + 1, dist_{x_8,y} + \sqrt{2}) \end{aligned} \tag{2}$$

[8] This distance is an approximation of the euclidean distance, that is computed faster.

x_1	x_2	x_3
x_4	x	x_5
x_6	x_7	x_8

Fig. 2. Neighbourhood of x

input image proximity image

Fig. 3. Example of a proximity image

The gradient direction for each point of the input image is also updated in order to give the direction of the closest contour point.

In the same time, the colour encoding of the input image (RGB) is transformed to Lab coordinates. In this colorimetric space, a skin colour model has been defined as a cylinder parallel to the luminance axis whose basis is circular in the space of chrominance (corresponding to an average range of chrominance coordinates of skins). This allows then to compute a low resolution binary image $IndC$, which reads which pixel (or corresponding square area of the input image) may correspond to a skin colour or not. Several cylinders can be defined according to different skin colours, in which case the $IndC$ binary image will indicate if the corresponding pixel area corresponds to at least one of the colour cylinders.

3.2 Genome and similarity criterion

The search space, i.e. the space of all possible solutions, is a 5 dimension space, which corresponds to the 5 following parameters:

- $model$ for the hand model (see figure 4)
- tx for the horizontal translation of the model,
- ty for the vertical translation,
- $scale$ for the apparent scale of the model, which varies with the size of the user's hand and its distance to the camera,
- $rotate$ for the rotation of the hand with respect to the optical axis of the camera.

The other rotations are ignored, as we consider that the user designates objects with arms stretched, and hands quasi othogonal to the optical axis. This allow to restrict the search to 2D models of hands.

The similarity criterion (or $fitness$) is a combination of three independent criteria, f_1, f_2 and f_3.

- f_1 gives a measure of the distance between the image and model contours, it is based on the use of the proximity image (see figure 3):

$$f_1(S) = \frac{\sum_{p \in contour} p \times prox(proj(p))}{card(contour)} \tag{3}$$

  $S = [model, tx, ty, scale, rotate]$ is a point of the search space, $proj(p)$ is the projection of p via the transformation defined by S and $card(contour)$ is the number of contour pixels. Dividing by $card(contour)$ allows the criterion to be independent of the number of contour pixels.

– f_2 measures the correpondance of gradient directions between image area and model:

$$f_2(S) = \frac{card(C)}{card(contour)} \tag{4}$$

$card(C)$ is the number of contour pixels x whose direction corresponds to the model's direction, and $card(contour)$ is the number of pixels of the contour.

– f_3 uses the binary image $IndC$ to verify if the interior of models contours has the right colour (this calculation is usually done on a lower scale image):

$$f_3(S) = \frac{card(D)}{card(E)} \tag{5}$$

$card(D)$ is the number of interior pixels of the model whose projection on the image corresponds to "true" on $IndC$ and $card(E)$ is the number of interior pixels of the model. Once again dividing by $card(E)$ allows to have a measurement independent to the size of the model.

The resemblance criterion for a solution S is then:

$$fitness(S) = \frac{\alpha_1 \times f_1(S) + \alpha_2 \times f_2(s) + \alpha_3 \times f_3(S)}{\alpha_1 + \alpha_2 + \alpha_3} \tag{6}$$

In our implementation, we choose $\alpha_1 = 10$, $\alpha_2 = 10$ and $\alpha_3 = 1$.

3.3 Genetic engine

A steady state approach has been chosen for the EA engine, which means that each new individual immediately replaces an old unadapted one within the population: no generation synchronism is used. The individuals to be replaced are chosen using the tournament technique. Three genetic operators are used:

– *immigration:* a new individual is randomly generated,
– *mutation:* a new individual is a small perturbation of an existing one,
– *crossover:* a new individual is created as a random weighted average of two existing individuals.

After an immigration or a mutation is done, two individuals are chosen in the population and the one who gets the lower fitness is replaced with the new individual (tournament of size 2). After a crossover, a tournament of size 3 is used.

Steady state engine and tournament selection have been chosen in order to better maintain the diversity and limit the risk of premature convergence. This point is actually critical here, as we operate a small sized population in order to obtain real time performance. A generational process would necessitate fitness ranking which is time consuming. For the same reason, a fixed size population is used, in order to minimise time consuming memory allocations.

Additionnally, as this EA runs on an varying environment, the fitness function is recomputed at each new frame. We actually get an asynchronous algorithm, that uses video information at the time it is available.
Finally, to compare with a generational EA, we measure the evolution in terms of "pseudo-generations", which corresponds here to the evaluation of a number of individuals equivalent to a generation gap (i.e. 40% of the population size, see section 5).

4 Additional improvements

Experiments based on the previous algorithm yield good results, however in some conditions, some model representatives were lost in the population. As a consequence, the detection of a model change was unstable and slower. We thus preferred an encoding that naturally balances the number of representatives within the population, by not explicitly using the $model$ any more: the fitness is then computed for each searched model. Each individual thus has now n fitness values, one for each of the n searched models. The resulting fitness is the maximal value of the n fitness:

$$fitness(i) = max(fitness_1(i), ..., fitness_n(i)) \tag{7}$$

$fitness(i)$ is the fitness value of individual i and $fitness_k(i)$ is the fitness value of individual i according to model k ($k \in [1..n]$), computed using equation 6.

For each frame, the detected model is the one which corresponds to the highest fitness, as soon as it is greater than a fixed threshold (0.3 for the results presented in the next section). To avoid unstabilities due to the definitions of a strict threshold, the detection decision is based on a double threshold and takes into account the model recognised at the previous frame:

- if the largest fitness of the last population of the current frame is larger than threshold T, the corresponding model is detected,
- if the largest fitness is in $[0.7 \times T, T]$, and corresponds to the detected model of the previous frame, it is maintained as "detected,"
- else, nothing is detected.

5 Results

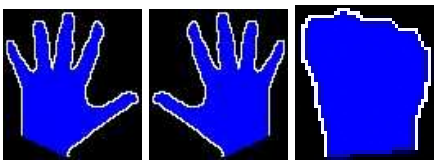
The results presented below have been obtained on a netBook *MSI U100*, equipped with a single core 1,60 GHz *Intel Atom N270* processor, with 1 GB of memory, and using *GNU/Linux Mandriva 2009*. The images are captured with the integrated webcam (resolution of 160 $\times$ 120 pixels).

Examples of detection on a video sequence are displayed on figure 6, and the curves of figure 5 give a representation of the distribution of representatives of each of the searched models within the population. Each graph (one per searched model, see figure 4) presents three curves which are the maximal and average of fitness values, and the proportion of individuals whose fitness is at least equal to 90% of the maximal fitness. The horizontal line represents the model detection threshold (0.3). Vertical lines mark the transition between two successive frames. For each frame, its number, and the number of generations that were run on it, are displayed.

It can be noticed that the curve corresponding to the fitness max of model 1 (which is not present in this part of the video) never crosses the 0.3 threshold. On the contrary, one may notice on graphs of models 2 and 3, an alternance of detection of these models, and for each detection phase, a concentration of the population around the maximal values (green curve).

Other examples of detection are given on figure 7, in the experimental conditions described in section 1.1. A video sequence is also available on

`http://apis.saclay.inria.fr/twiki/bin/view/Apis/HandGestureRecognition`



Model 1 Model 2 Model 3

Fig. 4. Hand models: the white pixels are contours, blue pixels are pixels where the colour model is evaluated, and black pixels are ignored.

6 Conclusion

We showed that a real-time evolutionary algorithm can be run on a general-public device, using low computation power. The specific components of this evolution algorithm are a small population, an asynchronous steady-state genetic engine, that produces a variable number of generations between two frames, depending on the computation load of the whole system. An implementation on another computer would result in a different average number of generations between two frames. This characteristic yields an efficient capability to the algorithm, as it is able to exploit the available data as soon as they appear. Of course the number of generations between two frames condition the quality of the detection, and its reactivity to sudden changes. In extreme conditions, if the algorithm cannot obtain enough computation power, the population does not have enough time to converge between two frames. From the user's side, this results in late detections, and unability to follow rapid moves.

The genetic encoding finally adopted (section 4) has been preferred for robustness and reactivity reasons. In case the genome carries the model, a delay is necessary in order to let representatives of a new model grow, while in the second version, this change can be immediate. Of course this solution is only efficient if the number of searched models is low, as fitness calculation is longer (n fitness computations for n models).

Additionnally, for the application described in section 1.1, a Kalman filter on the detected parameters for each frame allows to obtain a smoother detection. Finally, the display of a "mouse pointer" in the virtual environment (yellow mark in figure 7) gives functionalities similar to classical computer mouses.

Acknowledgments: The authors thank Sergio Alejandro Mota-Gutierrez, of Guanajuato University, Mexico, student of the *Electrical Engineering Master Program*, for his contribution to the colour based fitness computation.

References

1. Darby, J., Li, B., Costen, N.: Activity classification for interactive game interfaces. Int. J. Comput. Games Technol. **2008** (2008) 1–7
2. Kim, H.J., Kwak, K.C., Lee, J.: Bimanual Hand Tracking. In: Computational Science and Its Applications - ICCSA 2006. Volume 3980/2006. Springer (2006) 955–963
3. Carbini, S., Viallet, J.E., Bernier, O.: Pointing gesture visual recognition by body feature detection and tracking. In: COMPUTATIONAL IMAGING AND VISION. Volume 32., Kluwer Academic Publishers (2006) 203–208

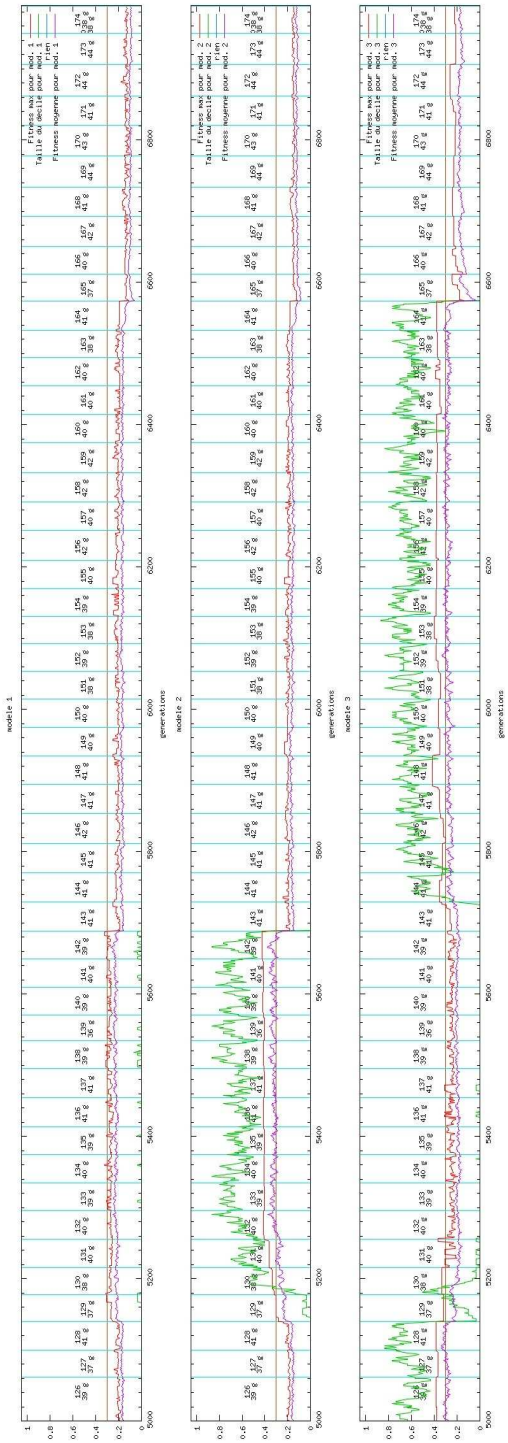


Fig. 5. Variation of the population models distribution and fitness values, for a sequence where the 3 models of figure 4 are searched.

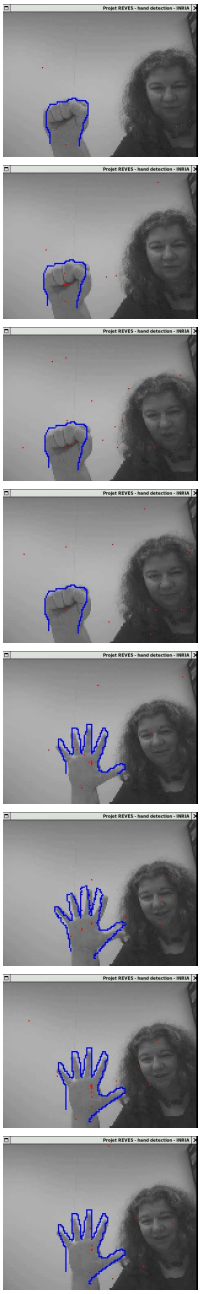


Fig. 6. Example of detection on a video sequence in natural light environment. Blue shapes show the detected model, red points correspond to the position (origin of the models) (x, y) of all individuals of the current population.

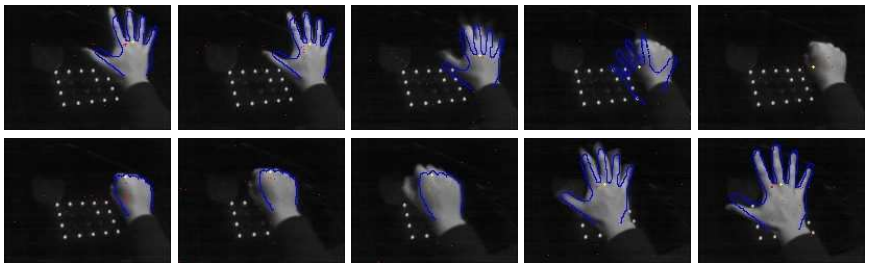


Fig. 7. Example of a detection on a video sequence in controlled light environment. A yellow marker represents the mouse position.

4. Maimon, D., Yeshurun, Y.: Hand Detection by Direct Convexity Estimation. In: Advanced Studies in Biometrics. Volume 3161/2005. Springer Berlin / Heidelberg (2005)
5. Ionescu, B., Coquin, D., Lambert, P., Buzuloiu, V.: Dynamic hand gesture recognition using the skeleton of the hand. EURASIP J. Appl. Signal Process. **2005** (2005) 2101–2109
6. Zhao, S., Tan, W., Wen, S., Liu, Y.: An improved algorithm of hand gesture recognition under intricate background. In: ICIRA '08: Proceedings of the First International Conference on Intelligent Robotics and Applications, Berlin, Heidelberg, Springer-Verlag (2008) 786–794
7. Harper, R.H.R., Rauterberg, M., Combetto, M., eds.: Entertainment Computing - ICEC 2006, 5th International Conference, Cambridge, UK, September 20-22, 2006, Proceedings. In Harper, R.H.R., Rauterberg, M., Combetto, M., eds.: ICEC. Volume 4161 of Lecture Notes in Computer Science., Springer (2006)
8. Stødle, D., Hagen, T.M.S., Bjørndalen, J.M., Anshus, O.J.: Gesture-based, touch-free multi-user gaming on wall-sized, high-resolution tiled displays. Journal of Virtual Reality and Broadcasting **5**(10) (November 2008)
9. Kirishima, T., Manabe, Y., Sato, K., Chihara, K.: Real-time multiview recognition of human gestures by distributed image processing. J. Image Video Process. **2010** (2010) 1–13
10. Hough, P.V.C.: Methods and means of recognizing complex patterns. Technical report, US Patent 3.069.654.18 (December 1962)
11. Maître, H.: Un panorama de la transformation de Hough. Traitement du Signal **2**(4) (1985) 305–317
12. Lutton, E., Martinez, P.: A Genetic Algorithm for the Detection of 2D Geometric Primitives in Images. In: 12-ICPR. (1994) Jerusalem, Israel, 9-13 October.
13. Louchet, J.: From Hough to Darwin: an Individual Evolutionary Strategy applied to Artificial Vision. In: Artificial Evolution, European Conference, AE 99, Dunkerque, France, November 1999, Selected papers. Volume LNCS 1829. Springer Verlag (1999)
14. Rechenberg, I.: Evolution Strategy : Nature's way of optimization, H. W. Bergman (Ed). In: Optimization : methods and and applications. Possibilities and Limitations. Volume 17. Springer, Berlin (June 1989) 106–126 Lecture Notes in Engeneering.
15. Baeck, T., Hoffmeister, F., Schwefel, H.P.: A survey of evolution strategies. In: International Conference on Genetic Algorithms. (1991) 2–10 13-16 July.
16. Goldberg, D.A.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, inc., Reading, MA (January 1989)
17. Borgefors, G.: Distance transformations in arbitrary dimensions. Computer Vision, Graphics, and Image Processing **27** (1984) 321–345