Interactive Evolution of Images

Jeanine Graf and Wolfgang Banzhaf

Abstract

Systems of selection and variation by recombination and/or mutation can be used to evolve images for computer graphics and animation. Interactive evolution can be used to direct the development of favorite designs in various application areas. Examples of the application of evolutionary algorithms to two-dimensional (2-D) bitmap images and the methods for three-dimensional (3-D) voxel images are indicated. We show that artificial evolution can serve as a useful tool for achieving flexibility and complexity in image design with only a moderate amount of user-input and detailed knowledge.

1 INTRODUCTION

Dawkins (Dawkins 1986) demonstrated convincingly the potential of Darwinian variation and selection in graphics. He evolved *biomorphs* 2-D graphic objects, from a collection of genetic parameters with interaction with the user (Dawkins 1986; Smith 1984). Recently, much research has been directed into the application of genetic algorithms to image and graphics problems, such as the segmentation of range images (Meygret, Levine, and Roth 1992) or pattern identification (Hill and Taylor 1992). Sims (Sims 1991) used genetic algorithms for interactive generation of color art; Todd and Latham (Todd and Latham 1991) have considered similar ideas to reproduce computer sculptures through structural geometric techniques. Caldwell and Johnson (Caldwell and Johnston 1991) have applied the concept of genetic algorithms to search interactively in face space of criminal suspects with the help of witnesses .

The novel idea offered in this paper is to provide a user with new technique to evolve 2-D (bitmap) and 3-D (voxel) images that can be applied universally in any field of interest. We have developed a system which presents progressively evolving solutions for graphical design problems by means of interactive processes.

Interactive evolution is a technique from the class of evolutionary algorithms (EAs) which are based upon a simple model of organic evolution. Most of these algorithms operate on a population of individuals which represent search points in the space of the decision variables (either directly, or by using coding mappings). Evolution proceeds from generation to generation by exchanging genetic material between individuals (*recombination*), i.e. by trying out new combinations of partial solutions, and by random changes of individuals (*mutation*). New variations are subjected to *selection* based on an evaluation of features of the individuals according to certain (*fitness*) criteria.

The best-known representatives of this class of algorithms are evolutionary programming (EP), developed in the U.S. by L.J. Fogel (Fogel, Owens, and Walsh 1966), evolution strategies (ESs), developed in Germany by I. Rechenberg (Rechenberg 1973) and H.–P. Schwefel (Schwefel 1981), and genetic algorithms (GAs), developed in the U.S. by J.H. Holland (Holland 1975).

Evolutionary programming (EP) tries to apply the variation-selection principle to ameliorate computer systems. The important and elementary steps of the search process are (i) a definition of the task and its fitness criteria, (ii) creation of a first representation, (iii) variation of the statement leading to offspring representations, (iv) performance qualification of the offspring representations, (v) reproduction of the best performing representations. The process is repeated until the task is accomplished (Fogel 1993).

Evolutionary programming uses real-valued object variables and normally distributed random mutation with expectation zero. The variance of the distribution evolves during the optimization process. It is calculated separately for each representation as transformation of its own fitness value. Mutation is the primary operator. Recombination is not used in the standard EP algorithm. Objective function values are scaled and possibly randomly altered to obtain the fitness. Selection of new parent representations is done probabilistically using a competition procedure which guarantees that the best representation is always retained and the worst always discarded.

The first applications of *evolution strategies* (ES) came in the field of experimental optimization and used discrete mutations. When computers became available, algorithms were devised that operated with continuous-valued variables.

The ES uses, like EP, mutation as its main search operator. In addition recombination operators (e.g. *discrete recombination*) are applied. Parameters like the standard deviation of the mutation are added to the representation (individual) as *strategy parameters*, which are adapted during the simulation via heuristics like Rechenberg's $1/5$ *success rule* (Rechenberg 1973). Only the μ best individuals out of the λ offsprings or out of the offsprings *and* parents are selected to form the next population (Schwefel 1981; Bäck 1994; Rechenberg 1973).

The *genetic algorithm* (GA) works on a genotypic level of binary encoded individuals, a choice that is founded by the argument of maximizing the number of schemata available with respect to a given code (Goldberg 1989; Davis 1991). Various selection schemes, such as proportional selection, are applied with respect to the relative fitness of the individuals. The recombination (e.g., 1-point crossover) serves as the main search operator. Mutation (e.g., bit-mutation) is used at a low rate to maintain diversity. Nearly no knowledge about the properties of the object function is required. In order to use a GA for optimization, a mapping from the genotype (bitstring) to phenotype (realised behavior) has to be defined. This could be a very complicated task, because the mapping is absolutely crucial for the performance of the GA.

In all of these cases, the selection criteria are traditionally fixed and are held constant from the start of the simulation, therefore these criteria must be detailed explicitly beforehand. This constitutes a significant problem in many realistic applications (apart from optimization), because an explicit fitness function may not be available in closed form. Recently, various work-arounds have been tried, one of the most prominent being *co-evolution*. In this method, rather than using one population to search for the best solution, two or more antagonistic populations are run which compete against each other. The realized fitness in this case is in part determined by the relationship of one population to the other, and does not have to be defined explicitly beforehand (Hillis 1990).

This paper makes use of an alternative method to generate fitness by involving the user into the selection process of artificial evolution. Our work in computer graphics, a natural domain for humans, easily engages the user by relying on human visual capacity.

2 INTERACTIVE EVOLUTION

In *interactive evolution*, the user selects one or more favourite model(s) which survive(s) and reproduce(s) (with variation) to constitute a new generation. These techniques can be applied to the production of computer graphics, animation, creating forms, textures, and motions (Glassner 1990; Arvo 1991; Foley 1992). Potential applications of interactive evolution include product design, e.g., cars, engineering of components, and architectural design.

Phenotypes and Genotypes

We shall need to discern between genotypes and phenotypes in interactive evolution, both terms are also basic concepts for biological evolution. The biological genotype is the information that codes the development of an individual. Genotypes most often consist of DNA sequences. In interactive evolution, genotypes are represented as numerical data and real values, collections of procedural parameters, symbolic expressions or compound data structures (e.g., trees). The phenotype is the realised behavior of the individual itself, i.e., the product of an interpretation of the underlying genotypic representation. In our case, the phenotype is the resulting graphical image.

The relation between genotypes and phenotypes in nature is determined by the genotype-phenotype mapping. This transformation is very complicated and draws heavily from the individual's current environment. In principle, a similar mapping can be introduced in artificial evolution (Banzhaf 1994). In this way, a part of the complexity of the developing solution might be rendered by the environment.

Fitness and Selection

The term fitness in interactive evolution is the capability of an individual or model to survive into the next generation, and therefore is tied directly to selection. Usually, fitness is not defined explicitly but is instead a relative measure of success following from the selection activity of a human user. Here, it is even based on non-quantifiable measures like visual preference or personal taste.

Hybrid systems, however, are reasonable as well. Certain predefined criteria (for example: drag coefficient (c_w) or air resistance of an airplane model) help to sort candidates for survival from the set of all variants, among which a human user finally selects the next generation.

Variation

In interactive evolution, one of the main benefits is the automatic generation of variants. Variation is accomplished by defining problem-specific mutation and recombination operators that constantly propose new variants to the presently existing population of graphic models on the screen. A certain amount of knowledge has to be invested in order to find appropriate operators for an application domain. In the next section we shall provide appropriate operators for manipulation of bitmap images and voxel graphics.

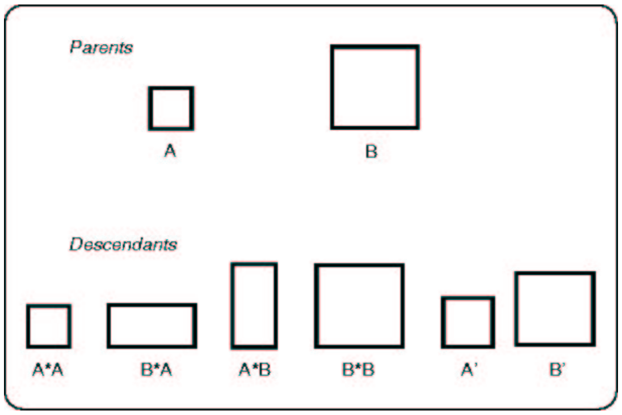

Figure 1: Generation of variants by recombination and mutation of graphical models. A*A and B*B are the unchanged parents, B*A and A*B are generated by recombination, and A′ and B′ are generated by mutation

3 EVOLVING 2-D IMAGES

In our approach we try to evolve *two-dimensional* images specified either directly as bitmaps or as parameterized geometric models, such as those provided by vector graphics. Figure 1 gives a sketch of the variants that may arise from the graphical parent objects. Whereas the latter is more or less straight-forward in EAs, once the parameters have been fixed, the former is a challenging task. To our knowledge no effort has been made to apply evolutionary algorithms to the evolution of *bitmap* images directly.

Application to 2-D Images

Bitmaps and other forms of direct encoding of images have found an excellent niche in computer graphics, video composition and image rendering (Foley 1992; Kirik 1992). Any 2-D shape can be represented as a sequence of points or vertices, with each vertex consisting of an ordered pair of numbers (x, y), its coordinates. The array of pixel values of a 2-D image, however, have nothing to do with the structure being represented in the image. This constitutes the challenge to finding appropriate operators for the generation of new variants of an existing image, because structural or functional conservation of the image content is of utmost importance in application.

We solve the problem for realizing evolutionary operations by using *warping* and *morphing* to create variations.

Warping is a method which, by using tiepoints in two images "A" and "B", allows for the creation of intermediate images (Woldberg 1990; Ruprecht 1994). Basically, these intermediate images are interpolations along an abstract axis from image "A" and image "B". The tiepoints are constraints of the interpolation because corresponding tiepoints in "A" have to be transformed into those of "B".

Morphing is an application of digital image warping. It involves distributing tiepoints over two images in such a way as to conserve essential structures in interpolated (intermediate) images. In this way an arbitrary initial image can evolve via intermediate steps of interpolation into a final image without leaving visual irritations.

It is interesting to note that even without any concept of structure in image warping and morphing, visually appealing images are generated. Structure has been substituted by tiepoints that extend their influence into the surrounding image by the help of interpolation algorithms.

We adopt this novel approach for the artificial evolution of images. By specifying tiepoints, sufficient control can be exerted about structure in 2-D images as to provide useful variants to the images being varied. Whereas in conventional computer graphics morphing is used for the purpose of transforming images in a dynamic animation series, we use the generated intermediate images as variants of the original images in the process of evolution.

Let us look more closely at the operations usually implemented in EAs:

Mutation is commonly considered to be a local operation that does not radically change the resulting phenotype. In order to provide this feature in bitmap image evolution, we propose to use a very small number of tiepoints. A one-point mutation would select one tiepoint in an image. The corresponding tiepoint in a second image would then be used as a source for novelty, by providing information into which direction to evolve the original image. Structure is conserved because tiepoints in both images correspond to each other. A parameter would then be used to quantify the degree of substitution in the image.

Note that the second image, from which novelty is gained, is not necessarily in the present generation of the evolutionary process. Instead, a generation 0 of images, equipped with a number of tiepoints is used for mutation. By selecting one tiepoint in an image of generation *n* that corresponds to a tiepoint in an image of generation 0 and constraining the effect to a local neighborhood, we provide a path for morphing between the two images. The generation 0 images in a way help to form equivalence classes between structures expressed as tiepoints. Some domain knowledge must be used in the process of tiepoint selection for generation 0.

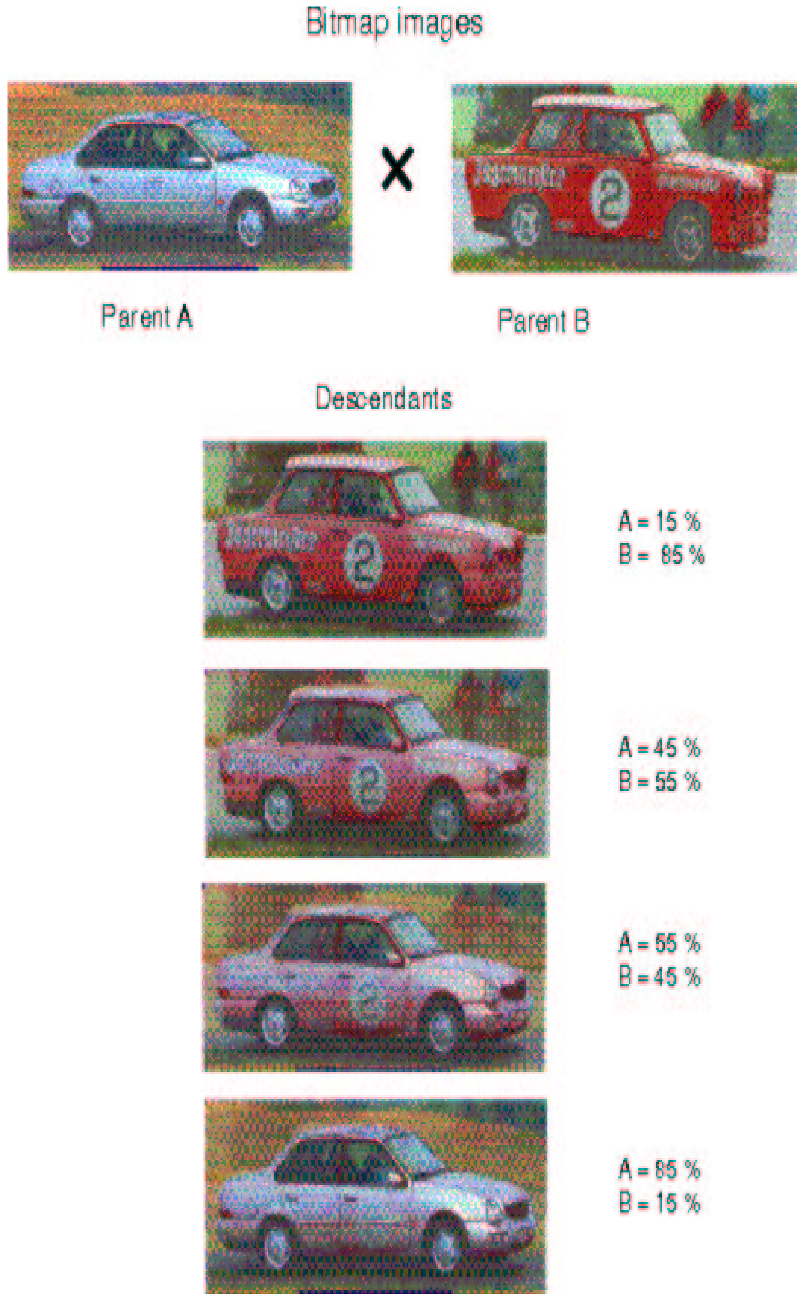

Figure 2: This figure shows an example recombination of two bitmap images through image interpolation. The percentage represents the proportion of inheritance from the parent images

Recombination is implemented as a more global operation by which two images exchange information. We propose to use as many tiepoints as necessary to conserve the underlying structure in two images “A” and “B”. A recombination would then be quantified in the image space between “A” and “B” by a certain parameter indicating the degree of “intermediateness” of a variant. Figure 2 demonstrates the method of recombination. Different variants between the two original cars are shown. Note that recombination always operates within the present generation.

Figure 3 and 4 demonstrates the mutation process by using 2-D images of cars. A local variation takes place by substituting one sort of wheel by another (Figure 3).

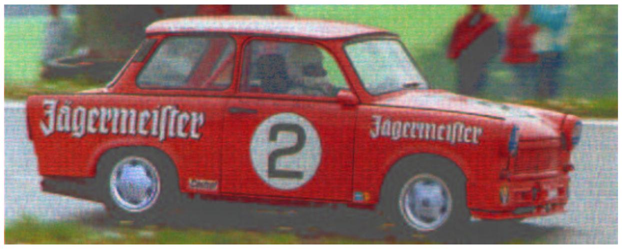

Figure 3: Local mutation by substituting one wheel by another.

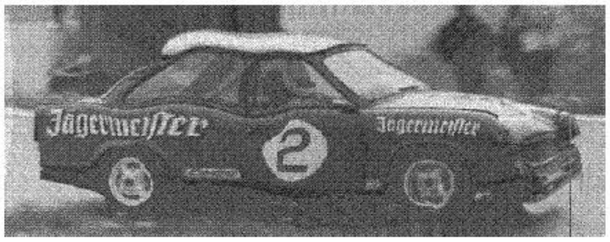

Figure 4: Whole mutation of a car by slightly random warping.

An arbitrary warping of an image at different locations, without using the proper equivalence class of image structures is shown in a contrasting image in Figure 4. It gives the impression of a damaged car. That is the case because arbitrary operations have been applied without being constrained by an otherwise existing path of variation between structures.

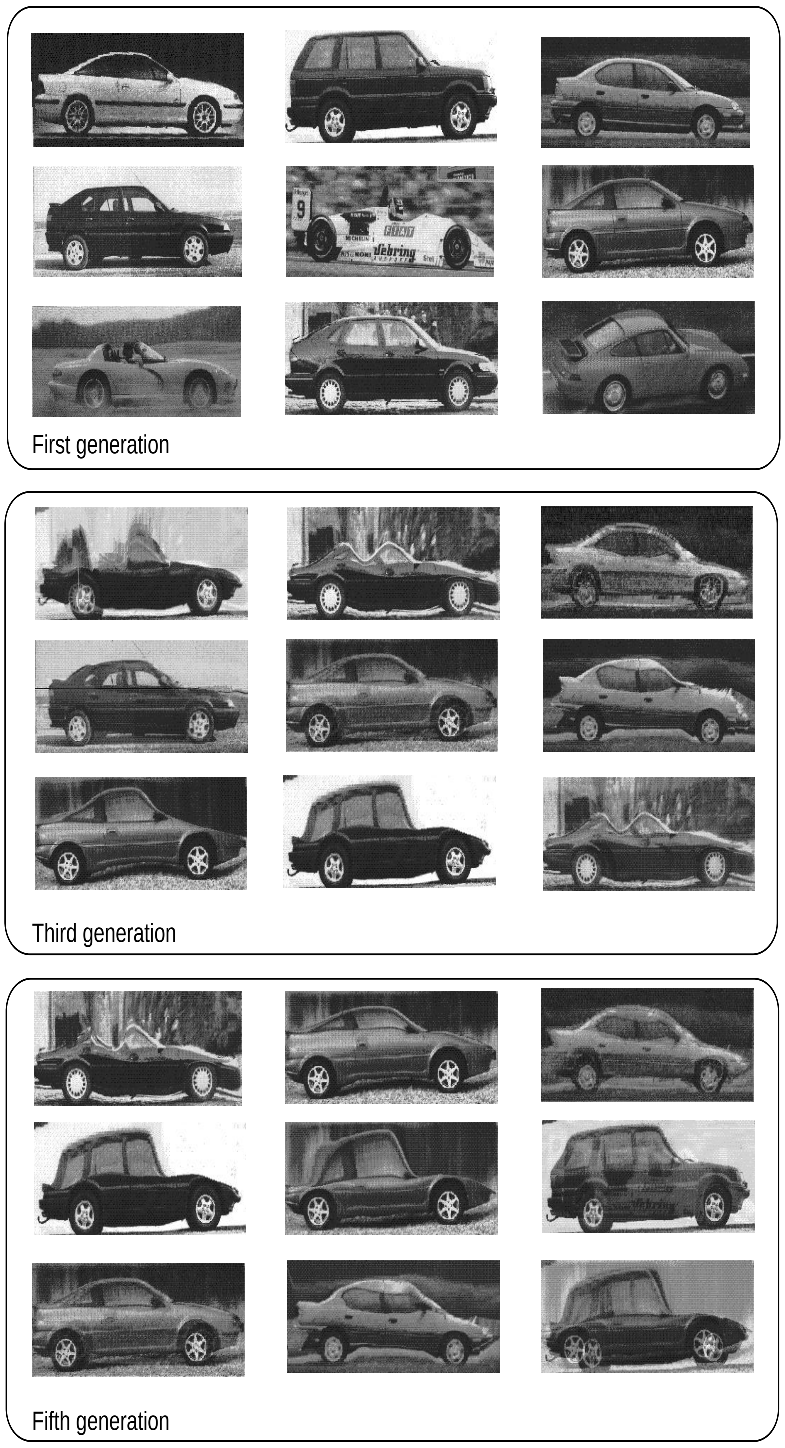

Figure 5: This example shows the evolution of nine models. After five generations some models are found which are closer to the users taste and to his target model.

Structurally, the content of an image is usually composed of components. In our example, a car is composed of a body, wheels, seats, chassis, engine, windows and doors. The same method that was applied before to the entire image representing the whole structure can also be applied to its components. By using many tiepoints in a component such as, say a wheel, influence can be exerted to any necessary degree about the details of the evolving structure.

Before presenting the entire interactive evolution system, we now turn to other representations of images.

Application to 2-D structural descriptions of images

Procedural models of images can be characterized by certain parameters that must be interpreted in the appropriate context. The parameters constitute the genotype of an image. Its interpretation is the genotype-phenotype mapping and the resulting image is the phenotype.

Because the number of structural elements usually varies from image to image, it is necessary to allow for variable-length genotypes.

Variation takes place on the level of parameters that are subjected to normally distributed random mutations as well as to intermediate or discrete recombination operations.

The resulting images are subjected to the same selection procedures as are those of the bitmap manipulation procedures discussed before.

4 EVOLVING 3-D IMAGES

For various applications in computer graphics it is often useful to represent objects in a 3-D grid of cubes or *voxels* (volume elements) according to their position in space (Young and Pew 1992).

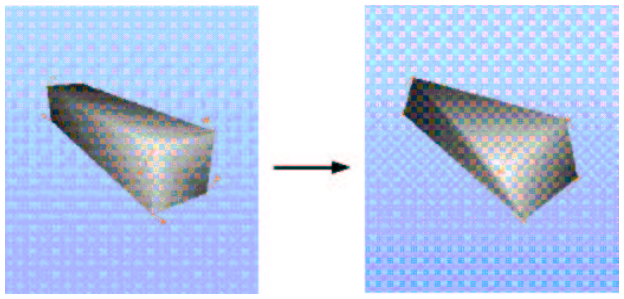

Figure 6: 3-D mutation

From objects in 3-D space an image can be constructed using established methods of computer graphics (Watt 1993). Alternatively, procedural models of 3-D objects can be combined into an image.

The generalisation of the above methods into the realm of 3-D graphics is straight-forward. Tiepoints in 3-D voxel graphics influence 3-D areas instead of 2-D areas in pixel graphics. Apart from that, any other component of the mechanism remains in place, especially interpolation and the generation of local and global variation to an original parent generation of images.

The same applies to procedural models that are generated by applying a genotype-phenotype mapping starting from a collection of appropriate parameters. Figure 5 and 6 show two examples of 3-D mutations that can be used.

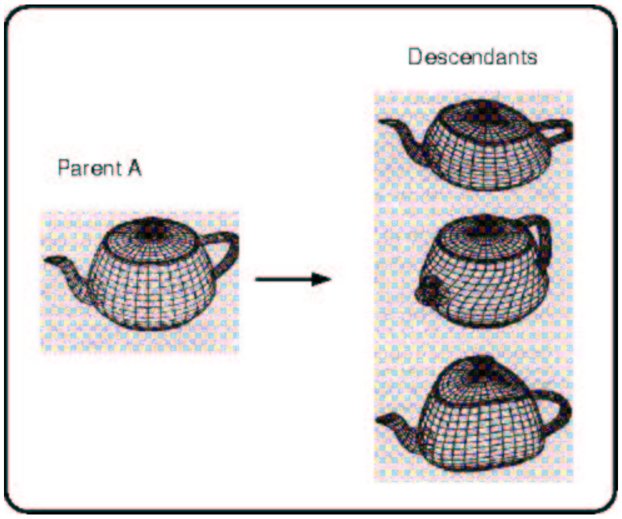

Figure 7: Mutation through deformation [26]. A set of transformations that deform the object. Linear transformation rotate, translate or scale the object.

5 JARDIN – A SYSTEM FOR INTERACTIVE EVOLUTION

Jardin is a digital image warping and morphing program, that allows the evolution of images, and runs under the X Window System (Cutler, Gilly, and O'Reilly 1993; Gaskin 1992; Young and Pew 1992). Jardin loads and saves image populations. It provides facilities to store tiepoints in images, to warp images and to apply the evolutionary process. Tiepoints are inherited from generation to generation, with generation "0" provided by the user. With a very small population, between 2 and 20 graphical models per generation and over a short time, a human user can select new generations of images. This process will be repeated until a favourite individual in the population has been generated.

6 SUMMARY AND CONCLUSIONS

We demonstrate how interactive evolution can be applied to 2-D bitmap images and a generalization to 3-D representation is outlined. The main idea is to combine the concepts from interactive evolutionary algorithms with the concepts of warping and morphing from computer graphics. Structure within images is substituted by a collection of tiepoints. By providing a first generation of images, where a structure in the images can manually be defined by the user. Evolution then proceeds along the paths constrained by the set of these tiepoints in all of the images. The original generation is kept as a source for mutations, which allows for new models to be created. In contrast recombination always works on images of the present generation.

In our version of *interactive evolution*, the user selects his favourite individual which then is reproduced to constitute the next generation. These techniques can be applied to the production of computer graphics and include e.g., product visualisation of cars, planes, engineering components and construction projects. Our interactive evolution system has the potential for a large number of other application areas like: interactive plotting in business, electronic publishing, computer aided design, drafting and manufacturing, simulation and animation for scientific visualisation, entertainment, architecture, etc. Our interactive simulations have shown that interesting results can be achieved even with low population sizes and few generations. This makes the system applicable to quick design and prototyping, in a large variety of application areas.

Acknowledgements

Funding from the German Bundesministerium für Forschung und Technologie (BMFT) under project EVOALG is gratefully acknowledged. Thanks to Detlef Ruprecht for help and morphing graphics software support. Thanks to Thomas Bäck for discussions. And special thanks to David Fogel for his helpful comments and suggestions.

References

Arvo, J. (1991). *Graphics Gems*. California: Academic Press.

Bäck, T. (1994, February). *Evolutionary Algorithms in Theory and Practice*. Doctoral dissertation, University of Dortmund.

Banzhaf, W. (1994). The genotype - phenotype mapping - a case study in genetic programming. In Y. Davidor, H. P. Schwefel, and R. Männer (Eds.), *Proc. PPSN-94 Jerusalem*, Springer Berlin.

Caldwell, C. and V. Johnston (1991). Tracking a criminal suspect through face-space with a genetic algorithm.

Cutler, E., D. Gilly, and T. O'Reilly (1993, sep). *The X Window System in a Nutshell*. The Definitive Guides to the X Window System. Sebastopol, CA: O'Reilly & Associates, Inc.

Davis, L. (1991). *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinold.

Dawkins, R. (1986). *The Blind Watchmaker*. Longman, Harlow.

Fogel, D. (1993). On the philosophical differences between evolutionary algorithms and genetic algorithms. In D. Fogel and W. Atmar (Eds.), *Proceedings of the Second Annual Conference on Evolutionary Programming*, Evol. Prog. Soc., La Jolla, CA, pp. 23–29.

Fogel, L. J., A. J. Owens, and M. J. Walsh (1966). *Artificial Intelligence through Simulated Evolution*. New York: Wiley.

Foley, T. A. (1992). *Computer Graphics Principles and Practice*. Addison-Wesley.

Gaskin, T. (1992). *DGXWS - PEXlib Programming Manual 3D Programming in X*. Sebastopol, CA: O'Reilly & Associates.

Glassner, A. (1990). *Graphics Gems*. California: Academic Press.

Goldberg, D. E. (1989). *Genetic Algorithms in search, optimization and machine learning*. Addison-Wesley.

Hill, A. and C. Taylor (1992, June). Model based image interpretation using genetic algorithms. In *Image and Vision Computing, vol.10*, pp. 295–300.

Hillis, W. D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. In *Physica D42*, pp. 228–234.

Holland, J. (1975). *Adaption in Natural and Artificial Systems*. The University of Michigan Press: Ann Arbor.

Kirik, D. (1992). *Graphics Gems III*. California: Academic Press.

Meygret, A., D. Levine, and G. Roth (1992). Robust primitive extraction in a range image. In *Conf. on Pattern Recognition, Vol. III*, pp. 193–196.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Frommann–Holzboog.

Ruprecht, D. (1994, May). *Geometrische Deformationen als Werkzeug in der graphischen Datenverarbeitung*. Doctoral dissertation, University of Dortmund.

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models*. Chichester: Wiley.

Sims, K. (1991, Jul). Artificial evolution for computer graphics. In *Computer Graphics, Vol.25*, pp. 319–328.

Smith, J. (1984, July). Plants, fractals, and formal languages. In *Computer Graphics, Vol.18, No.3*, pp. 1–10.

Todd, S. and W. Latham (1991). *Mutator, a Subjective Human Interface for Evolution of Computer Sculptures*. IBM United Kingdom Scientific Center Report.

Watt, A. (1993). *3D Computer Graphics* (second ed.). Reading, Massachusetts: Addison-Wesley.

Woldberg, G. (1990). *Digital Image Warping*. IEEE Computer Society Press.

Young, D. and J. Pew (1992). *The X Window System Programming & Applications with Xt*. Englewood Cliffs, NJ: Prentice Hall.