Connection Science, Vol. 6, Nos. 2 & 3, 1994

Towards Automated Artificial Evolution for Computer-generated Images

SHUMEET BALUJA, DEAN POMERLEAU & TODD JOCHEM

In 1991, Karl Sims presented work on artificial evolution in which he used genetic algorithms to evolve complex structures for use in computer-generated images and animations. The evolution of the computer-generated images progressed from simple, randomly generated shapes to interesting images which the users created interactively. The evolution advanced under the constant guidance and supervision of the user. This paper describes attempts to automate the process of image evolution through the use of artificial neural networks. The central objective of this study is to learn the user's preferences, and to apply this knowledge to evolve aesthetically pleasing images which are similar to those evolved through interactive sessions with the user. This paper presents a detailed performance analysis of both the successes and shortcomings encountered in the use of five artificial neural network architectures. Further possibilities for improving the performance of a fully automated system are also discussed.

KEYWORDS: Artificial neural networks, computer-generated images, genetic algorithms, genetic programming.

1. Introduction

In automating a system to produce aesthetically pleasing images, two fundamental components must interact. The first component encompasses the mechanisms to create images. The second component must evaluate the images and choose the next move. The system developed here draws from the field of genetic algorithms and genetic programming for the mechanisms used to create potentially pleasing images. The tool used to evaluate the images produced by the genetic procedures is an artificial neural network.

Genetic algorithms were chosen as the method for creating images because they are general-purpose tools designed to explore irregular, poorly characterized function spaces. One such function space is the space of possible pixel images. With the aid of a genetic algorithm, a user can explore the space of images. This exploration, although dependent upon the user, is aided by the mechanisms inherent to the genetic algorithm. Karl Sims showed this to be a very effective method of creating aesthetically pleasing images (Sims, 1991). In viewing genetic algorithms as a search

S. Baluja, D. Pomerleau and T. Jochem, School of Computer Science, Department of Computer Science, Camegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA. E-mail: baluja@cs.cmu.edu, pomerleau@cs.cmu.edu and tjochem@ri.cmu.edu.

tool, perhaps a better explanation of the role of the genetic algorithm is to emphasize their ability to 'find' appealing images rather than to create appealing images. This distinction will become apparent throughout this paper. The goal of this work is to automate the process of finding aesthetically pleasing images—to simulate the constant interaction of the user.

Artificial neural networks (ANNs) are the principal tool used in automating the evaluation of images. There were several reasons for choosing ANNs over other potential methods for addressing this problem. Perhaps the most important motivation was that the desired computation can be 'programmed' into the ANN by repeatedly presenting examples of the desired behavior. This property allows ANNs to perform tasks for which creating explicit rules may either be an overwhelming or an extremely time-consuming endeavor. This property also maps well to the requirement of the task. It is often the case that a user will be able to decide whether or not a particular image is appealing, but will find it difficult to express what particular characteristic, or set of characteristics, makes it appealing. For many rule-based systems, this constraint would make the task of automating the user's role extremely difficult. However, as ANNs only require examples of desired behavior, this small amount of information may be enough. This ability must also be considered in light of the drawbacks it presents. If the ANN is able to perform the task on which it is trained, in other than the simplest of tasks, it becomes a very difficult undertaking to determine what the ANN has 'learned', and an even harder task to translate the knowledge embedded in the ANN into understandable rules.

In order to ensure that there is as little bias as possible in the ANN, the raw two-dimensional image is used as the input into the ANN. This ensures that the ANN is not limited to the features which the experimenters deem to be important; rather, the ANN can develop the features necessary for accomplishing the task.

The remainder of this section gives an introduction to image evolution. In Section 2, image evolution is explored in depth, through the perspective of the genetic algorithm. Section 3 describes how the user can create images using the genetic algorithm's search mechanisms. Section 4 describes several attempts to automate the user's role with artificial neural networks. Section 5 compares the ANN architectures explored in Section 4 on a sample test set. Section 6 describes how the trained ANNs can be used to automate image evolution, and presents six automatically evolved images. Finally, Section 7 presents methods of improving the automation of future systems.

1.1. Image Evolution

In 1991, Sims presented a novel approach for combining genetic algorithms with computer graphics (Sims, 1991). The system which Sims designed allows users to evolve complex figures without concern for the mathematics used to generate the images. The interface is simple: given a number of initially random figures, the users select the two which are the most interesting. These figures are used as 'parents' to produce a subsequent population of 'offspring'. The offspring possess some attributes of both parent images. From the new population of images, two parents are selected, and the cycle continues. Through this iterated process of interactive selection, the images, which may have started out as simple, uninteresting lines, can become interesting images which the users have evolved under their guidance.

Sims extended his work beyond the production of figures; he also explored the evolution of solid textures, three-dimensional plant structures and animations. One of the most attractive aspects of this style of interactive graphics development is that it abstracts many of the cumbersome details of image production away from the users. The users are not required to know how the graphics are generated, how offspring are produced from two parent images or how the images are internally represented.

The power of this method lies in the ability to direct the progress of evolution. Perhaps the easiest way to describe both how and why this process works is by analogy. By selecting some images to be the parents of the next generation, and not selecting others, the users create a bias in the evolution based upon their own likes and dislikes. The figures which are 'strong', with respect to the users' tastes, are more likely to be selected as the parents of the next generation. Assuming that the parent 'chromosomes' (the internal representation of the images) have a means to pass their 'qualities' to their children, the characteristics found in the parent images are also found in varying degrees in members of the subsequent populations. Continuing this analogy, Darwin's theory of survival of the fittest plays an integral role in explaining how subsequent populations become closer to the users' preferences. The users' preferences are the basis of the fitness function. Through a number of generations, the characteristics which the users do not find interesting will not be selected for recombination. Only the images which contain characteristics which the users find interesting will be selected; therefore, only these will influence the composition of subsequent populations.

The image evolution employed in this study, and in Sims' work, uses symbolic expressions in prefix form to specify images. These expressions specify how to calculate a color value for each pixel coordinate on a two-dimensional plane. Upon selection of two parent images, the images are recombined through the use of genetic cross-over and mutation operators. Details of these operators and the primitives which comprise the defining expressions are given in Section 2. From the two parents, a new population of children is produced, and the cycle is continued.

1.2. ***Background*** *Information*

One of the most attractive features of the type of evolution described in the previous section is the simplicity of the users' role; users only have to select the images which they find interesting. Although Sims used this basic idea for evolution in a variety of domains, the scope of this study is limited to two-dimensional images. The ideas presented here can be extended to the other domains examined by Sims, as the principles underlying each are very similar.

The central objective of this study is to learn the user's preferences and to apply this knowledge to evolve images which are similar to those evolved through interactive sessions with the user. In the system described to this point, the user must remain an active participant throughout the entire evolution, continuously selecting two images to be the parents of the subsequent generation's population. To attempt to learn a user's preferences, the user is asked to rate interest in each image in a set of collected sample images. These ratings are used to train an artificial neural network, which receives as training examples an image and the user's rating of the image. It is hoped that the artificial neural network will be able to generalize user preferences to images not in the initial sample, thereby making possible an automation of the selection process. This has the potential to reduce participation of a user to only judging a small set of images, rather than supervising the entire evolution.

2. Simulating Evolution with Genetic Algorithms

2.1. An Introduction to Genetic Algorithms

The genetic algorithm (GA) is established upon the foundations of natural selection and genetic recombination. A GA combines the principles of survival of the fittest with a randomized information exchange (Goldberg, 1989). Although the information exchange is randomized, the GA is far different from a simple random walk. A GA has the ability to recognize trends towards optimal solutions, and to exploit such information by guiding the search toward them.

A genetic algorithm maintains a population of potential solutions to the objective function being optimized. The initial group of potential solutions is determined randomly, These potential solutions, called 'chromosomes', are allowed to evolve over a number of generations. At every generation, the fitness of each chromosome is calculated. The fitness is a measure of how well the potential solution optimizes the objective function. The subsequent generation is created by recombining pairs of chromosomes in the current generation. Recombination between two chromosomes is the method through which the populations 'evolve' better solutions. The solutions are probabilistically chosen for recombination based upon their fitness. Although the chromosomes with high fitness values will have a higher probability of being selected for recombination than those which do not, they are not guaranteed *to* appear in the next generation. The 'children' chromosomes produced by the genetic recombination are not necessarily better than their 'parent' chromosomes. Nevertheless, because of the selective pressure applied through a number of generations, the overall trend is towards better chromosomes.

In most applications which employ genetic algorithms, the objective function is well defined. However, for this application it is not easy to define a clear objective. Although it is simple to say that the objective is to search for 'interesting' shapes and figures, this leads to further complications. Not only does 'interesting' vary from person to person, it also vanes in an individual from instance *to* instance. In this system, the user has an integral role in deciding which images are interesting, and therefore serves the role of the objective function. This will be explored in greater detail in Section **2.4.** Once the user has supplied the evaluations, the genetic algorithm can proceed in the same way as a standard genetic algorithm using a clearly defined objective function.

In order to perform extensive search, genetic diversity must be maintained. When diversity is lost, it is possible for the GA to settle into a local optimum. The basic GA uses *two* fundamental mechanisms to maintain diversity. The first, mentioned above, is a probabilistic scheme for selecting which chromosomes to recombine. This ensures that information other than that which is represented in the best chromosomes appears in the subsequent generation. Recombining only good chromosomes will cause the population to converge very quickly without extensive exploration, thereby increasing the possibility of finding only a local optimum. The second mechanism is mutation, which helps to preserve diversity and to escape from local optima. Mutations introduce random changes into the population.

The GA is typically allowed to continue for a fixed number of generations. At the conclusion of the specified number of generations, the best chromosome in the final population, or the best chromosome ever found, is returned. Unlike the majority of other search heuristics, GAs do not work from a single point in the function space. Many methods which use only a single point to explore the function space are very susceptible to local optima. GAs continually maintain a population of points from which the function space is explored; this aids in finding global optima (Baluja, 1992).

2.2. The Composition of Chromosomes

In many optimization problems for which GAs are used, the chromosomes are represented as bit strings. Although such a representation is possible for this task, a more natural representation is symbolic prefix-order expressions. The set of 19 primitives from which the symbolic expressions are composed is shown below. The set of primitives used in this study is smaller and simpler than that chosen by Sims. If interesting images are to evolve, the GA must combine only these simple primitives in novel ways. The primitives are categorized into two classes, those which take one argument, and those which take two.

One argument:	reciprocal (1/argument), natural log, log (base 10), exponent, square, square root, sine, hyperbolic sine, cosine, hyperbolic cosine.
Two arguments:	average, minimum, maximum, addition, subtraction, multiplication, division, modulo, random (randomly choose either argument 1 or argument 2).

Using these simple primitives, a sample equation might be

subtract (log(x), avg(sqrt(y), log(x)))

where x and y represent coordinates on the pixel plane. One method of increasing the diversity in the images produced is to add new primitives. For example, with the primitives which Sims allowed, such as iterative statements, fractal images can also be evolved. Some of the other primitives which Sims included were noise generators, blur operators and bandpass convolutions. Several of these primitives rely upon neighboring pixel values; the primitives chosen for this study do not. Further, each of the primitives are functions of x and/or y only. For simplicity, no explicit constants are used in the equations. The next section will show how these equations are used in genetic recombination.

2.3. Recombination of Chromosomes

The effectiveness of GAs lies in their ability to recombine good solutions to produce potentially better solutions. As chromosomes are, in many GA tasks, represented as fixed-size bit strings, cross-over operators are usually straightfonvard: a randomly chosen section of bits from the two parent chromosomes are swapped, and the two resulting chromosomes are the offspring. However, as the potential 'solutions' used here are not bit strings, this simple form of cross-over is not appropriate for two reasons. First, restricting the growth of equations to a fixed size may be detrimental to the production of interesting images, as larger equations may result in more interesting images. Simple swapping may not work, as a long expression may not have corresponding counterparts in shorter expressions. Second, it is not evident how such a simple cross-over can be implemented, since two equations may have different structures, as shown in Figure 1.

In order to provide a general cross-over mechanism, a cross-over operator

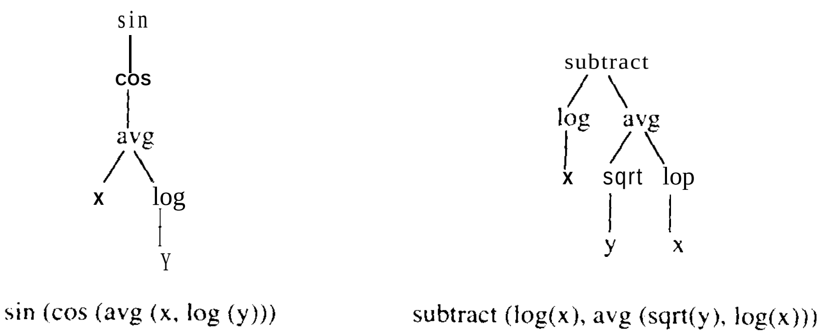

sin
cos
avg
x
log
Y
subtract
log
avg
x
sqrt
lop
y
x
sin (cos (avg (x, log (y)))
subtract (log(x), avg (sqrt(y), log(x)))

Figure 1. Two unique chromosomes structures which may be 'bred' together to produce 'children' chromosomes.

commonly employed in 'genetic programming' tasks is used (Koza, **1992).** It designates one parent to be a 'receiver', and the second to be a 'donator'. Once the roles have been assigned, the cross-over procedure is as follows.

(1) Randomly choose a node in the donator's tree structure.
(2) Randomly choose a node in the receiver's tree structure.
(3) Delete everything at, and below, the chosen node in the receiver's tree.
(4) Copy everything at and below the donator's chosen node to the receiver's chosen node.

Figure *2* shows five of the possible children of the *two* equations shown in Figure 1.

*2.4. Assessing **the Fitness of a** Chromosome*

The objective of the **GA** is to find 'interesting' shapes and figures. In order to avoid the complexities involved with attempting to quantify *the* quality of being 'interesting', the issue is not addressed directly. In this system, as well as Sims' system, the user plays an active role in deciding what is interesting: the user's

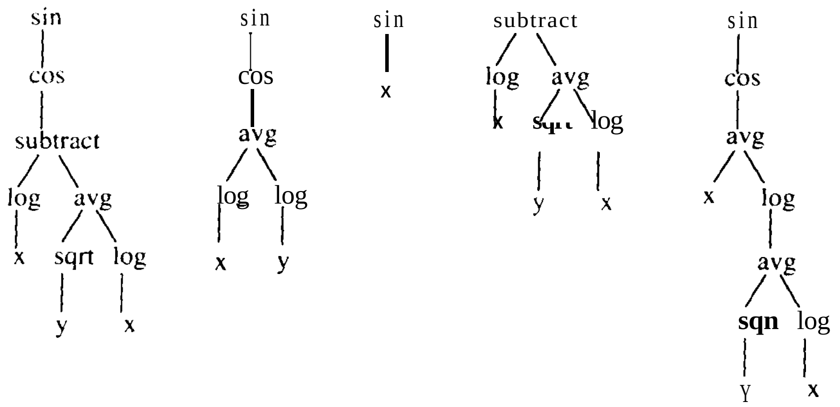


Figure 2. Five potential children of *sin(cos(avg(x,log(y))))* and ***subtract** (log(x), avg(sqrt(y),log(x))*. For these examples, the first equation is defined as the receiver and the second as donator.

preferences are the objective function. If an image in the current population satisfies a user's objective in evolving images, its chances of being chosen increase above the others. This not only avoids the problems associated with quantifying interest, it also does not limit the program to one interpretation of 'interesting'. The drawback to this method is that it requires the constant interaction of the user. In Section 4, methods for automating the interaction are presented.

2.5. The Need for Preserving Diversity in Genetic Search

Maintaining diversity is crucial to performing productive genetic search. If diversity is not maintained, premature convergence of genetic search can lead to non-optimal solutions. In the context of GAs used for function optimization, premature convergence can correspond to finding a strong local optima. In the context of image production, loss of diversity may cause the images produced after convergence to be very similar to each other. Unless the user is satisfied with the types of images produced, further attempts at search may be unproductive.

Three mechanisms are used to preserve diversity. The first is the alternating of roles between the two parents as receiver and donator. The second is inherent to the system: the users are free to choose images which are to their liking or which are simply 'different' from the other images on the screen. **As** a brief observation of users has shown, users will often choose an image because it is different from the other images on the screen. The third mechanism is mutation, which is described below.

2.5.1. Mutation. Recombining similar chromosomes over a number of generations can cause subsequent populations to converge very quickly. Mutation is a mechanism used to avoid stagnation in genetic search. In genetic algorithms which use chromosomes represented as bit strings, mutation is usually implemented as a random bit flip. In this system, the mutation operator selects a random node within the equation, and randomly changes its contents. If the function at the node has only one parameter, the new function is randomly chosen from the set of one-parameter functions, and similarly for two-parameter functions. If the node contains an 'x', it is changed to 'y', and vice versa. The mutation rate used in this study is a constant one mutation per chromosome per generation.

3. Interactively Evolving Computer-generated Images

The description of the user interface is not critical in understanding the methods used to automate the system. It is included to provide a description of the role of the user, and therefore of the task which will be automated.

3.1. The User Interface

When the program is initialized, nine images are displayed on the screen. These images correspond to nine randomly generated expression trees (chromosomes), using the primitives described in Section 2.2. Two of these images are selected by the user to be the parents of the next generation. From the chosen parents, nine new children are produced which replace the previous nine images displayed. This process is repeated until the user has found an image which is satisfactory.

The images are composed of 256 colors. There were two motivations for using only 256 colors. The first was that by the use of only a few colors, the emphasis in the images is shifted away from the colors in the image to the structures produced. Secondly, and more importantly, 256 colors allowed an easier implementation of the training phase of the *ANN*. This will be discussed in greater detail in Section **4**.

Unlike Sims' work, the possible colors were determined before the evolutionary process was started. However, this could easily be modified to allow the color selection to also be evolved. The color scheme chosen was 128 shades of grey, followed by 128 shades of red. We found that this gives pleasing results, partially because of the simplicity in the color scheme, and partially because it creates an artificial boundary between the bright white and very dark red colors.

3.1.1. ***Using the image repository.*** In addition to the nine images with which the user is presented, the user may, at any time, also store and recall images from the image repository. If the user decides that the population has become too homogeneous, as in Figure *3,* the user has the option to save the image repository to disk for later use, and to load previously saved images into the repository for immediate use. With this facility, it is possible to combine images from different sessions (see Figure **4).** Because of the random nature of genetic algorithms, the evolutions at each session will be different, and unique images will evolve. Using the image repository as a way to recombine previously stored images with the current images can be considered analogous to using several subpopulations to evolve chromosomes in parallel (Cohoon *et al.*, 1988).

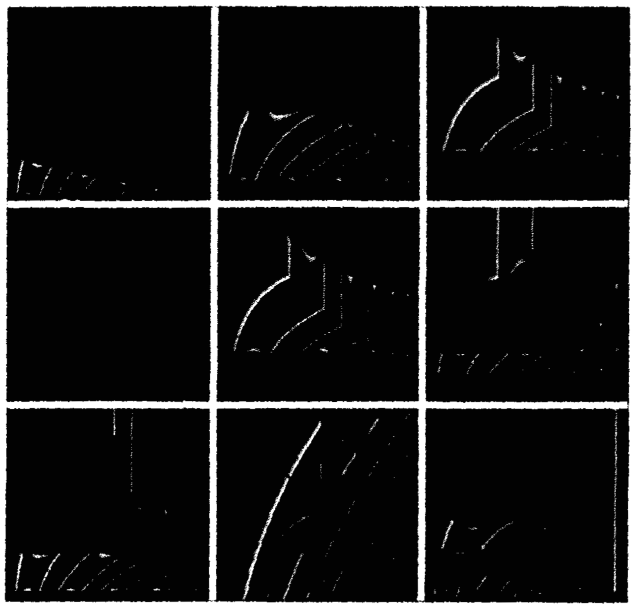


Figure 3. The nine images shown in this figure are children of the same parents. The images are very similar; heterogeneity in the population is lost. It is likely that any two parents chosen from this set of nine will lead to children which are very similar to the images seen here.

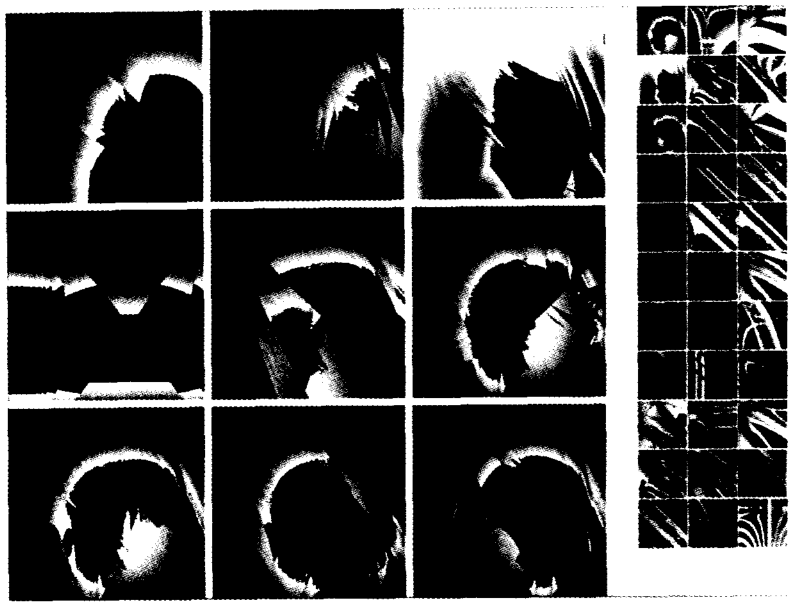


Figure 4. The user interface. The nine large images on the left represent the children of the previously selected parents. The images on the right are in the image repository. These images can be used for recombination in subsequent generations. The top three images in the first column of the repository are smaller copies of three of the images found in the larger squares. The defining equations for the three images on the bottom row of the nine large images are shown in the appendix.

3.2. Selected Images

Six images which were produced through several interactive evolution sessions are shown in Figure 5. Although these images used the red and grey color map described previously, the color map could have easily been chosen differently. For printing purposes, the images are presented in grey-scale.

4. Towards Automation

4.1. *Motivation and Overview*

To this point, the mechanisms for interactively evolving computer-generated images have been described. The users do not need to know any of the underlying mathematics or the mechanisms used to recombine the equations which the images represent. Although this provides a simple interface with which to interact, it is quite a challenging task to automate the user's decisions. The rest of this paper presents descriptions and analysis of the attempts to automate the process of parent selection, and the various successes and failures encountered in the pursuit of this goal.

The mechanism employed to learn the user's preferences is an *ANN*. The *ANN* is trained to give an evaluation to each image it is presented, without respect to any other image which may also be on the screen. The evaluation is on the scale of -1 to $+1$, with larger numbers representing greater preference. The network is

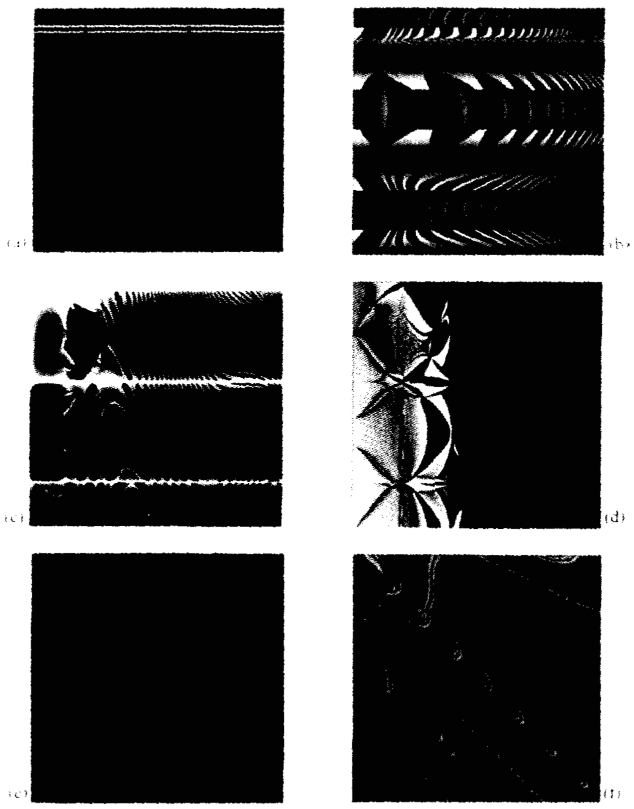


Figure 5. Six images, produced through several interactive evolution sessions, are shown in Figure 5. Although these images used the color map described previously, the color map could have easily been chosen differently.

trained on a large set of images which the user has previously graded on a similar scale.

ANNs were chosen as the learning tool because of their ability to generalize. If ANNs are not over-trained and are sufficiently large, they hold the potential for generalization to images which have not yet been encountered. Unlike many other non-learning techniques which may fail catastrophically if presented with images which have not yet been encountered, ANNs show a slower degradation of performance. The accuracy of the ANN's prediction degrades with the presentation of images which have decreasing similarity with the training images. Therefore, in training the ANN, it is important to get a diverse enough sample group to give a good representation of the input space. This is a particularly hard task for this domain, as the set of all possible functions which can be evolved is infinite.

The attempts to teach the ANN a user's preferences are conducted with raw two-dimensional pixels as the inputs to the ANN. The hope is that the ANN will develop the feature detectors and internal representations which are required to simulate accurately the preferences of the user. Pixel-based inputs have been used successfully in a variety of other tasks, such as autonomous road following (Pomerleau, 1992), gaze tracking (Baluja & Pomerleau, 1994) and recognizing handwritten ZIP codes (Le Cun *et al.*, 1989). However, in comparison to this task, the aforementioned applications have had a much more structured input. The variety of inputs the **ANN** is likely to encounter in this study is much larger than in the other domains; therefore, the internal representations needed may be substantially more complex.

One possible alternative to using only the image pixels is to pre-process the image with traditional vision-based techniques in order to extract features such as circles, lines, noise, etc., and use these high-level features as inputs into the **ANN.** However, the goal of this study is to perform the task with as little a *priori* knowledge of salient image features embedded in the *ANN* as possible. By building few assumptions about the basis of the user's preferences into the network, the system maintains the maximum amount of flexibility.

4.2. The Training Set

To train the *ANN,* a set of 400 unique 48 x 48 pixel images was used. The process used to obtain these images was as follows. Through several interactive evolutions, the two images the subject chose as parents of the next generation, as well as two randomly selected images from the selection of nine images present on the screen, became candidates for the training set. Several hundred images were collected in total. From this collection, 400 images were randomly chosen for the final training set. The subject was asked to rank these images individually on a scale of 0.1 to 1.0, in increments of 0.1, to gauge preference towards each image. To train the ANN, the subject's ranking was scaled to the range **–1** to +1. Although collecting only randomly generated images would have been easier than selecting images from interactive evolutions, randomly created images are generally not very interesting or pleasing. Therefore, in order to provide the *ANN* with examples of images which were ranked high as well as low, images which were selected by the user further into the interactive evolution were required.

The training set was developed using the above method to ensure that the included images uniformly spanned the set of possible rankings. The measures taken only partially accomplished this task. One of the difficulties inherent to the task of automation is that the relative availability of images which the user is likely to find uninteresting is very high in comparison to the images which the user is likely to find interesting. In order to show the distribution of the rankings of images obtained in a readily accessible format, the evaluations are divided into three categories, 'low', 'medium' and 'high'. The range of these categories is shown below. One of the peculiarities about this division is that although the 'low' region begins at 0.0, the users only graded images between 0.1 and 1.0, in increments of 0.1.

$0.0 \le \mathit{Low} \le 0.4$

$0.4 < \mathit{Medium} \le 0.7$

$0.7 < \mathit{High} \le 1.0$

Given this ranking, the distribution of the images into these three categories in the training and testing set is given in Table I. The majority of the images are classified by the user into the 'low' region. As will be seen in the next section, the skewed distribution of the images is a potential cause of degraded performance in classifying images in the 'medium' and 'high' ranges. In order to eliminate the effects of a training set with a skewed distribution of training points, the performance of ANNs

Table I. User classification of training and testing images

Image set	Number of images in set	Percentage of 'low' images	Percentage of 'medium' images	Percentage of 'high' images
Training	2000	64	19	17
Testing	1200	61	22	17

trained on uniformly distributed training sets was also measured. In several of the tests performed, a large number of the images ranked in the 'low' region were removed from the training set in order to make the number of images in each region more uniform. These tests will be described in greater detail throughout the remainder of this section.

A simplifying assumption made in the process of automation was that the user's preference of an image was based upon the structures present in the image. Further, it was assumed that the user would like the same structures in any degree of rotation. To achieve a small amount of rotation tolerance, each image in the training set was rotated three times by 90°. As the role of color was de-emphasized, while the role of shape was given more importance, the inverses of the original and each of the three rotated images were also captured. Because the color scheme only used 256 colors, the inverse of each image was calculated by subtracting the pixel's value from 256. This resulted in seven additional images for each original image; these additional images were assigned the same rating as the original image. Each image was composed of pixels with values in the range of 0–255, linearly scaled to values between –1 to +1. A total of 3200 images were produced in this manner. From the 3200 images, 2000 images were randomly chosen for use in training the ANN; the remaining 1200 images were used in the test set. It should be noted that the images within the training set and the testing set are correlated as they may be rotations or inverses of each other. No duplicate chromosomes (the equations which the image represent) were allowed in the set of 400 original chromosomes. However, because many images are taken from each sequence of evolutions, it is possible that different chromosomes may represent similar images. The ramifications of these limitations on the test set will be discussed in greater detail in Section 5.

4.3. *Preliminary Trials and Improvements*

The first ANN architecture explored was a simple one hidden-layer network, as shown in Figure 6. The input layer is arranged as a 48×48 grid, each coordinate corresponding to a pixel in the image. The input layer is fully connected to the hidden unit layer, which contains nine units. The hidden layer is fully connected to the output layer, which consists of a single output neuron. The output is a single value, which corresponded to the 'estimated preference value' of the image. The output neuron yields a value between –1 and +1, where –1 indicates a poor image according to its training set, and +1 indicates a good image. Although not shown, a bias input unit, a unit with its value permanently clamped to 1, is also present. The bias unit is connected to the hidden and output layers (Hertz *et al.*, 1991).

The single hidden-layer ANN, as well as all of the other ANNs mentioned throughout this paper, was trained using the standard error back-propagation algorithm. Although this network's error decreased rapidly through the training

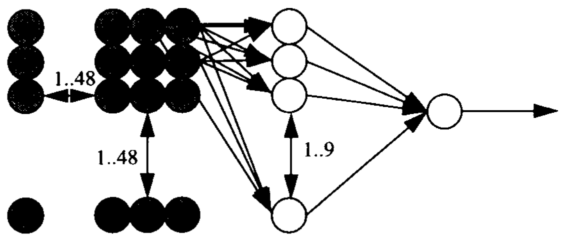


Figure 6. The first *ANN* architecture attempted. In this model, there are a total of ((48 x 48 + 1) x 9 + 9 x 1 + 1) = 20 755 connections, including the bias unit's connections.

process, the network was unable to make reasonable selections on the training or test sets. For example, the network sometimes chose almost blank images to be parent chromosomes. Although this sometimes led to interesting 'children', as blank images may be the result of complex equations, the choices did not correspond to the choices a human user would be likely to make. To improve the performance on subsequent attempts, both the network architecture and the presentation of the training and testing images was modified. The modifications are described below.

The first modification was to alter the red features to appear as grey. One difficulty with using pixel values as inputs into the *ANN* is that although the user sees a great distinction between the pixel values of 127 (white) and 128 (very dark red), the neural network cannot place the correct emphasis on this 'color cliff. The values which are given to train the neural network (127 and 128) are very close to each other, yet form a much more pronounced visual jump than the values from 126 to 127 or 128 to 129. The relative closeness of the color values does not adequately represent the great difference in actual appearance. To address this, all of the images were pre-processed to reduce both of the red and grey scales to only grey scale. This was accomplished by subtracting 128 from all pixels with a value greater than 127. Many of the important features of the image were still kept, although sudden discontinuities between bright red and bright grey in the original pictures were sometimes lost. However, the discontinuities which occurred between white and dark red were still prominent in the pre-processed image. This modification placed a larger emphasis on features rather than colors, and effectively reduced the color space to 128 shades of gray. Each pixel was again scaled to the range –1 to +1.

The second modification was the introduction of weight yoking, or weight sharing, in the *ANN,* to achieve some degree of shift invariance. Shift invariance is necessary, as a user who finds an interesting feature in one portion of the image may also find the same feature interesting in a different region of the image. Weight yoking is a technique used to find features regardless of where they appear in the image. LeCun *et al.* (1989) have used it successfully in their attempts to recognize hand-written Zip Codes. The technique of weight yoking in a spatial dimension is also commonly used outside of the visual domain. For example, similar techniques are frequently used in speech-recognition tasks (Waibel, 1990). As speech is fundamentally a process which occurs through time, one way of designing an *ANN* to learn patterns over time is to map the patterns into an ordered spatial dimension. As Todd points out, by performing this mapping, the problem of learning patterns over time becomes one of learning spatial patterns (Todd, **1989).** Although Todd opts for a different technique for using connectionist architectures to learn structures in music, he also points out that the problem is decomposable in a similar manner.

Training a standard *ANN* on pixel images will not encourage the recognition of features in locations other than those in which they appeared in the images of the training set. With weight yoking, groups of weights work as templates, or feature detectors, which are uniformly used in many locations throughout the input array. In networks that do not use weight sharing, when feature detectors are developed through training, they are only sensitive to locating features in the regions in which they occurred in the images of the training set. Weight sharing allows the developed feature detectors to be applied in many other locations. **A** more detailed description of weight yoking is given in LeCun *et al.* **(1989).**

Figure 7 provides a pictorial description of weight yoking. For this large

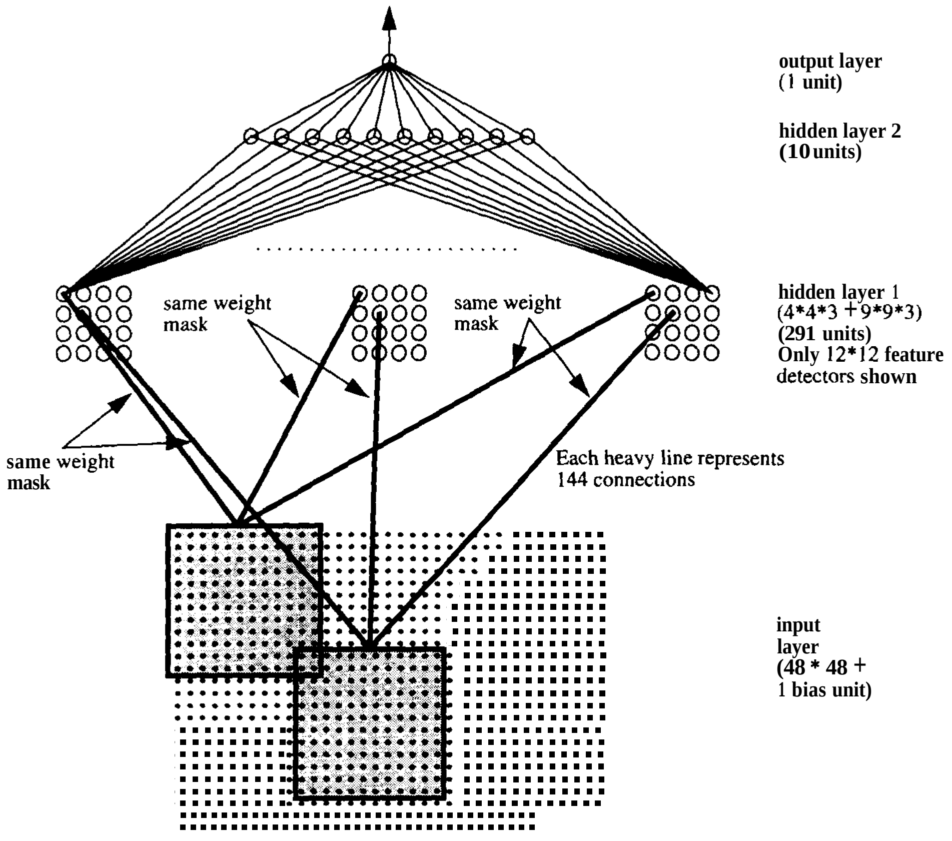


Figure 7. Only a small portion of the input layer is shown. There are a total of **9** x 9 (81) 6 x 6 input groups and a total of **4** x **4 (16) 12** x **12** input groups. Each 6 x 6 group is connected to three units in hidden layer **1.** Each **12** x **12** group is also connected to three units in hidden layer **1.** Only the **12** x **12** feature detectors are shown. The bias unit (not shown) is connected to each of the hidden units (in hidden layer **1** and in hidden layer **2)** and the output layer. The total number of connections is (81 x **(36 +** 1) x **3) + (16** x **(144 + 1)** x **3) + ((291 +** 1) x **10) +** (10+ **1)** = 18 **822.**

architecture, there were a total of six yoked groups, three which covered an area of the image 6 x 6 pixels in area, and three which were 12 x 12 pixels. Each yoked group has the same weight templates, which were applied to many regions across the image. For the tests presented here, the spacing between applications of the weight template was 5 x 5 pixels for the 6 x 6 groups, and 10 x 10 pixels for the 12 × 12 groups; the templates overlapped each other. The second hidden layer consisted of 10 fully connected neurons. The final output was a single unit.

4.4. Gauging Network Performance

The network was trained with the 2000-image training set previously described. In order to gauge the ability of the system to classify images as the user would classify them, a set of 1200 test images was used. Table II gives the total error in classifying the images. The error is computed as follows:

$$\mathit{Sum\ of\ Errors} = \sum_{i=0}^{1200} \left| (\mathit{UserPreference}_i - \mathit{NetworkEstimatedPreference}_i) \right|$$

User Preference and *NetworkEstimatedPreference* are normalized between 0 and 1.

The error in classifying the images decreases through the training period. Each epoch represents a presentation of the 2000 training images to the *ANN*. In order to create a point of reference from which to compare the performance of the *ANN*, four other simple measurements are given. Three of the measurements output constant values of 0.3, **0.5** and 0.7 for the evaluation of each image. The fourth measurement randomly selects a value between 0.0 and 1.0. Of these four measurements, the constant value of 0.3 should perform the best as the majority of the images were placed into the ‘low’ region by the user. Although the ANN classification’s were better than these four simple measurements, consistently producing an output of 0.3 does comparably well with the ANN outputs between 75 and 300 epochs.

In order to ensure that the *ANN* is learning more than the simple probability distribution of the user’s responses, the performance of a biased random number generator is examined. The random number generator was tested with two settings, the first which generates numbers with the distribution the *ANN* predicts at epoch 600, and the second in which the random numbers are generated in the actual distribution of the testing data. The results of this simulation revealed that the errors

Table II. Performance measurements—sum of errors (1200-image test set)

Method of classification	Sum of errors	Average error per image
ANN output (epoch 0)	371.71	0.31
ANN output (epoch 75)	298.00	0.25
ANN output (epoch 150)	288.74	0.24
ANN output (epoch 300)	287.50	0.24
ANN output (epoch 450)	271.58	0.23
ANN output (epoch 600)	269.78	0.22
Constant output of 0.30	292.90	0.24
Constant output of 0.50	324.50	0.27
Constant output of 0.70	443.90	0.37
Random output	402.94	0.34

Table III. Error per region—'Large' ANN architecture (1200-image test set)

	Error from classifying each region's images (% of total error)		
Description	Low	Medium	**High**
Random	64	17	19
Epoch *0*	78	05	17
Epoch 75	49	16	35
Epoch 150	50	15	35
Epoch 300	50	18	32
Epoch 450	48	22	30
Epoch 600	48	22	30

of using a biased random number output to be close to those of the ANN between epochs 0 and 75 (Baluja *et al.*, 1993).

In order to improve the performance of the *ANN,* it is first important to discover where the errors are occurring. Table **III** provides a breakdown of the occurrences of errors by region. During the beginning of training, at epoch 0, although the classifications are essentially random, they are largely within the medium range, as the network is initialized with intermediate range weights. **As** almost all of the images are classified within the 'medium' range, the images actually in the 'medium' range contribute very little to the total error. **As** training progresses, the performance on the **'low'** range images continues to improve at the expense of correct classification of both the 'medium' and 'high' range images. Because of the disproportionate number of images placed in the 'low' region by the user, the degradation in performance of *two* of the three regions still allows the overall classification error to be reduced. By epoch 600, the total error contributions of the 'low' region is 48%, and the 'high' region is 30%. However, the images ranked into the 'low' region by the user comprise 61% of the total number of images in the testing set, while the images in the 'high' region comprise only 17%.

Because of the skewed distribution of images in each region, correct classification of images in the 'low' region has the potential for greater error reduction than the correct classification of images in the 'high' range. During the early stages of training, the *ANN* reduces the total error by improving the classification of 'low' region images. After epoch **75,** the errors in classifying the images in the 'high' region are also reduced. **As** the initial classification placed all of the images into the 'medium' range, adding further discrimination decreases the tendency to classify images into this range, and thereby increases the errors associated with classifying 'medium' range images. Figure 8 shows graphically the average error of the three regions as a function of epochs.

In analyzing the distribution of correctly classified images, it was found that by epoch 600, 35% of all the images are classified within an error of 0.1. Only 26% of the images in the 'medium' range and **9%** of the images in the 'high' range fall into this category. The remaining images in this category constitute **45%** of the images in the 'low' range. It is evident, therefore, that if the test set included no 'low' images, the *ANN* would not perform well; the errors of the 'medium' and 'high' ranges would not be offset by the effects of correctly classifying 'low' images. **As** it is usually assumed that the training and testing sets have similar distributions, perhaps this result is expected.

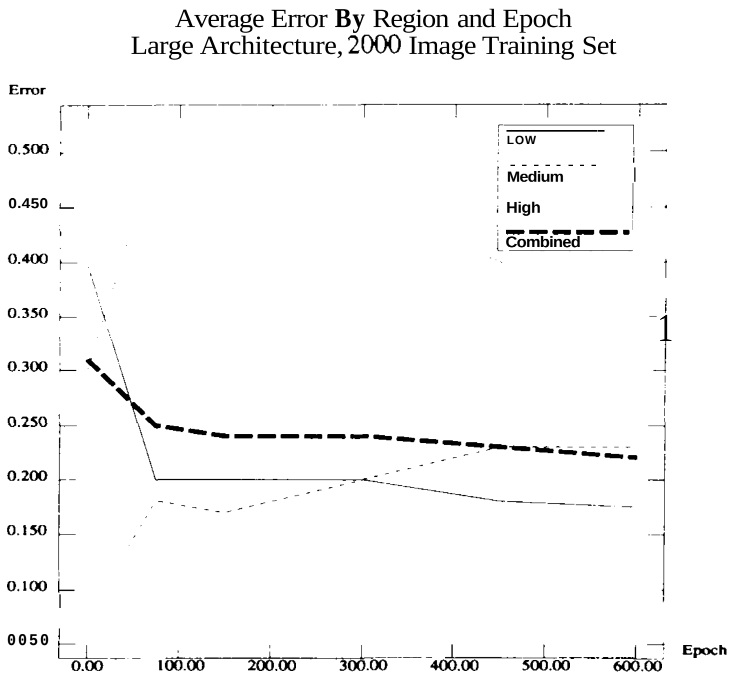


Figure 8. The average errors with classifying each region's images are shown as a function of epochs. The results for the 'Large' *ANN*, trained with the 2000-image training set, are shown. The performance is measured on the 1200-image test set.

The results given in this section accord very well with our intuitive expectations: training with a non-uniformly distributed training set will hurt performance in the classes which are not represented well. The question arises of what the *ANN* can learn from a uniformly distributed training set; one which contains approximately equal numbers of 'low', 'medium' and 'high' images. The first motivation for this test is determining whether the large number of training examples in the 'low' region made it possible to do well in classifying images in the 'low' region, or whether it was the features inherent to the images classified within the 'low' region. The second motivation is the hope that once the *ANN* is trained on a uniformly distributed training set, there will be less reliance on the distribution of images, and the *ANN* will be able to perform well on uniformly distributed test sets, as well as non-uniformly distributed test sets, which are likely to be generated by human users.

In order to rectify the problems associated with a skewed distribution of images in the training set, a smaller, uniformly distributed training set was created. This smaller training set comprised 1120 images. Of the 2000 images which were present in the previous training set, 880 randomly selected 'low' ranked images were removed. After removal of these images, the distribution of the smaller training set was 'low': 36%, 'medium': 33%, 'high': 31%. In order to gauge performance to the previous results, the same test set, of 1200 images, was used.

The first problem encountered with a reduced-size training set was that the ANN described in the previous section was able to memorize a large portion of the training data, thereby making generalization impossible. To reduce the probability of memorization, three variations of reduced-size *ANNs* were tested. Each is based

Table IV. Original and three smaller ANN architectures

Architecture	12 x 12 retina grids	6 x 6 retina grids	Total hidden units	Total connections
Original, Large ANN	3	3	301	18882
Small *ANN* 1	3	0	58	7461
Small *ANN* 2	1	1	107	6308
Small ANN 3	0	2	172	7635

upon the yoked-weight architecture described previously, and shown in Figure **7.** The primary difference between these architectures and the previously described 'Large' *ANN* is a reduction in the number of feature detectors. The difference between the three smaller architectures is the number and types of feature detectors each contains. **As** only the size and number of feature detectors vary in the three small *ANNs*, the performance of each architecture will reflect the usefulness of its feature detectors in this domain. The number of connections was kept approximately equal in each of three smaller architectures. Table IV gives a brief description of the three smaller *ANN* architectures.

Each architecture was trained on the 1120-image set multiple times, with slightly different parameter settings. The performance of Small *ANN* 3 on both the training and testing sets was much worse than that of Small *ANN* 1 and 2. The failure of Small *ANN* 3 lends support to the hypothesis that the ***6*** x ***6*** retina grids are too small to pick up important features. It should also be considered that when the ***6*** x ***6*** retina grid **is** used in conjunction to the 12 x 12 retina grids (Small *ANN* 2), the performance of the network was comparable to using additional 12 x 12 grids (Small *ANN* 1). Due to Small *ANN* ***3's*** inability to match the performance of the other *ANNs*, this architecture is not explored further in this paper.

The performances of Small *ANN* 1 and 2, trained with the 1120-image set, were comparable. **A** detailed examination of Small *ANN* 1 is presented in the next section. In the interest of space, the same detail is not devoted to Small *ANN* 2; however, it should be noted that the behaviors of both networks were very similar. The performance of Small *ANN* 2 is briefly returned to in Section 5; more details on this architecture can be found in Baluja ***et al.*** (1993).

4.5. *Using* ***a Smaller Architecture: Small ANN 1***

This *ANN* architecture consisted of three 12 x 12 yoked groups. The performance of this *ANN,* trained on the 1120-image set, is measured on the 1200-image test

Table V. Distribution of error—Small ANN 1 (1200-image test set)

Epoch	Summed error	Error from classifying each region's images (% of total error) Low	Medium	High
0	340.52	73	14	13
75	343.28	73	05	22
150	298.58	67	14	20
300	279.69	64	**19**	17
600	287.83	62	21	17
1200	297.67	61	22	17

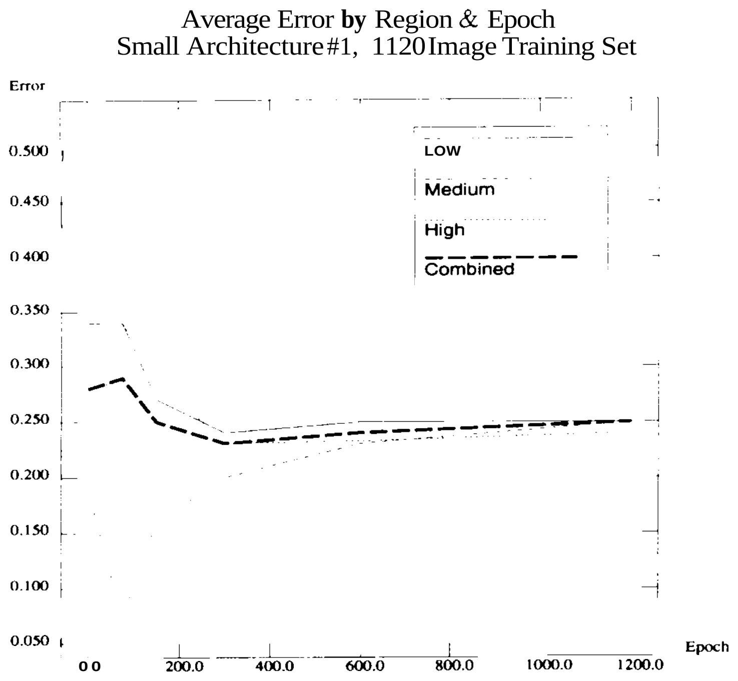


Figure 9. The average errors with classifying each region's images are shown as a function of epochs. The results for Small *ANN* 1, trained with the 1120-image training set, are shown. The performance is measured on the 1200-image test set.

set. Table **V** shows the distribution of errors at six points during training. By epoch 300, the error contribution from each region closely matches the percentage of images actually in each region of the test set. Unlike the original architecture examined, there is no bias towards better classification of 'low' images. Figure **9** shoes graphically the average errors for the three regions as a function of epochs.

In order to complete the comparison between the smaller architectures described above with that of the original 'Large' *ANN* described earlier in this section, one additional test needs to be performed. In the original test performed with the 'Large' *ANN,* there were more feature detectors and a larger training set than in the tests performed with Small *ANN* 1. In order to discern which of these two differences accounts for the difference in performances, Small *ANN* 1 needs to be trained with the larger, 2000 image, training set. It should be noted that the difference in performance was small, the 'Large' *ANN* was able to decrease the error to 269.78 (Table II), and the Small *ANN* 1 to 279.69 (Table V).

In this test, the 2000-image training set used was the same biased one used to train the large network. The testing set used is the same 1200 images used for testing the previous architectures. The lowest error reached with Small *ANN* 1 was at epoch 600. The summed error was 224.30, considerably lower than the lowest error, 269.78, achieved by the large network (Table 11). This error is also smaller than the lowest errors achieved by any of the smaller architectures using a uniformly distributed training set.

The distribution of the errors among the three regions was most similar to the original, large, architecture. The distribution of errors across the regions was: 'low':

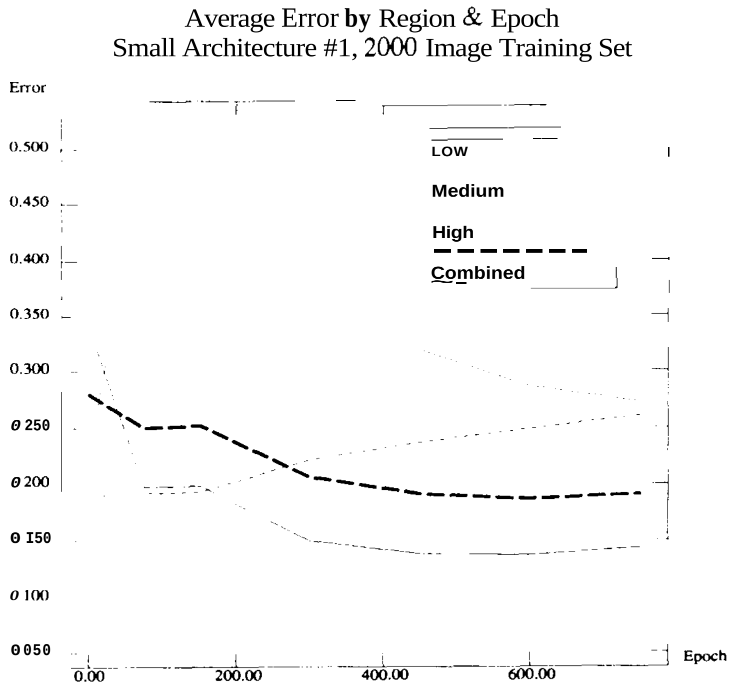


Figure 10. The average errors with classifying each region's images are shown as a function of epochs. The results for Small *ANN* 1, trained with the 2000-image training set, are shown. The performance is measured on the 1200-image test set.

47%, 'medium':29%, 'high': 26%. The skewed distribution of training images had the same effect on this architecture as the original 'Large' *ANN;* the images in the 'low' region were classified more accurately than those in the other regions. Figure 10 shows the average error of the three regions, using the Small *ANN* 1 and the full 2000-image training set, as a function of epochs. The average errors in classifications for Small *ANN* 1 were slightly higher with the larger training set than with the small training set for the 'medium' and 'high' range images (Figure 9, epoch 300 & Figure 10, epoch 600). However, the classification of 'low' images was more accurate using the larger training set. In comparison with the 'Large' *ANN,* which was also trained with the 2000-image set, this architecture classified the images more accurately. This indicates that this architecture may be more suited to this task than the original 'Large' *ANN.*

Figure 11 shows the distribution of the ANN's predicted classification and the human user's actual classification for the 1200-image test set. While the network underestimates the number of images in the lowest bin, the overall distribution matches the human's fairly closely.

There are at least three potential reasons why the presentation of the 2000-image training set leads to improved performance in comparison to the 1120-image training set. The first is the most intuitive—by giving an *ANN* more examples to train with, the *ANN* will be able to provide more accurate classifications. The second is that the correlation between the training and testing sets has increased. By examining the overlap between the 2000-image training set with the 1200-image

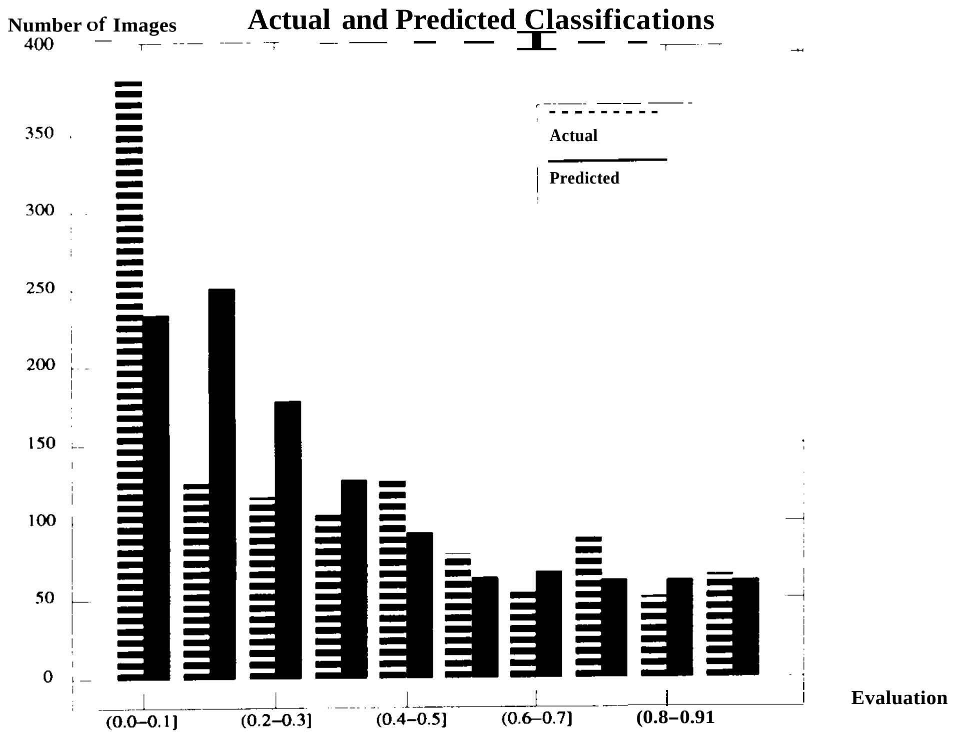


Figure 11. A histogram showing the distribution of responses from the user and the network on the 1200-image test set. The network's responses are taken from epoch 600.

testing set, 3.4% of the images were found in both the test set and the training set. In the smaller training set, consisting of 1120 images, 3.0% of the images in the test set were in the training set. Although there is an increase in the number of identical images in the larger training set, the increase is very small. The overlap in the training and testing sets occurs because different defining equations for the images may represent the same pixel image. In addition, because the images in the training and testing sets may be rotations or inverses of each other, the extra 880 images in the larger training set provide a much fuller, and more correlated, training set for the test set used. The third contributing factor is that by providing the network with many more 'low' images, the relative importance of correctly classify images in the 'low' region increases above the correct classification of images in the other two regions. In order to determine which of these factors contributed to the improved performance, an additional test was conducted with a new test set. The results are reported in the next section.

5. A Comparison of Architectures

In order to eliminate some of the biases in the training and testing sets mentioned in the previous section, an entirely new test set was gathered. It was chosen in a similar manner to the previous set, through several interactive evolution sessions. The new test set comprised a total of 100 images. From the set of 100 images, the

Table VI. Performance of *ANNs* on test set 2 (100-image test set)

ANN architecture	Number of image in training set	Summed error	Distribution of error (% of total) Low	Medium	High
Large *ANN*	2000	34.05	63.3	32.2	4.4
Small *ANN* 1	1120	23.09	49.5	41.5	9.0
Small *ANN* 2	1120	20.99	47.8	43.9	8.2
Small *ANN* 1	2000	20.50	47.5	47.2	5.3
Random	N/A	31.64	68.8	28.8	2.3
Biased Random	N/A	24.28	53.7	42.0	4.2

user ranked 64 in the 'low' region, 32 in the 'medium' region and 4 in the 'high' region. The performance of the architectures which have performed well on the 1200-image test was measured at the epoch in which their error was minimized for the 1200-image test set. The performance of these architectures on the 100-image test set is shown in Table VI. Also shown is the performance of random classifications and biased random classifications. The biased random classifier outputs a classification in the 'low', 'medium' and 'high' regions with probabilities 0.64, 0.32 and 0.04, respectively. Although the results of the biased random classifier are included for comparison, the exact distribution of images would, of course, not be available when using the *ANN* to simulate the human user.

The results in Table VI are somewhat disappointing. The 'Large' ANN, which had delivered good results on the original 1200-image test set, did significantly worse than all the smaller *ANN* architectures. Further, it did worse than both the random and biased random outputs. Due to the correlation between training and testing sets, perhaps the most likely explanation of the success of the 'Large' ANN on the original testing set may be its greater potential for memorization of features required to do well in both the training and testing sets.

The best result came from Small *ANN* 1 and 2. Small *ANN* 1, with the large training set, and Small *ANN* 2, with the small training set, performed comparably. It is suspected that a larger training set could have also improved the performance of Small ANN 2. On the 100-image test set, both smaller architectures were able to classify images within the 'low' region much more accurately than the other regions, whether trained with the original training set or with the smaller, uniformly distributed training set.

In the previous section, it was shown that training Small *ANN* 1 with the larger set resulted in improved performance when tested on the 1200-image test set. This trend is again seen here with the 100-image test set. However, the results reported also suggest that the disproportionately large number of 'low' images in the training set may not be the only reason which makes them easier to classify (Table VI). Perhaps another reason for the more accurate classification of images in the 'low' range is that a large portion of the 'low' region's images may be similar. These images may be blank, or very simple horizontal or vertical lines. The possible uniformity and simplicity of the features contained in images which are classified by the user as 'low' may make them easier for the *ANN* to identify accurately. In contrast, attempting to classify images in the 'medium' and 'high' ranges is much more difficult as this requires a much deeper knowledge of the preferences of the user.

6. Discussion and Pictorial Results

Many factors make the process of automation difficult. First, there is the large task of collecting a diverse set of images from all the regions on the evaluation scale. This is an inherent difficulty for this task, as most images produced will probably not be interesting. It is only through the process of evolution that interesting images are found. A possible solution, which was explored in this paper, is to collect images while repeatedly simulating the process of manual evolution. The importance of using images from several evolution sessions is two-fold. First, using evolutions, rather than randomly generated images, increases the chances of obtaining a larger range in the user's rankings of images. Second, using several sessions ensures that the images produced will be diverse, as each simulation evolves unique images.

Another factor which makes this problem difficult is the desire to accomplish the task while embedding as little *a priori* knowledge as possible of the features which contribute to a user's preferences. Although some knowledge is inherent in an ANN architecture which implements weight sharing, the amount of knowledge is kept relatively small. The space in which this work was done, that of two-dimensional pixel images, was certainly not the easiest with which to train an ANN. Using two-dimensional pixels made the size of the networks very large, as multiple connections are constructed for each input pixel. Further, as the 'image space' defined by pixels is very large in comparison to the number of samples obtainable, generalization based upon the pixel representation is difficult. Other feature spaces could have been chosen, in which the input to the network is not the pixel values, but a more abstract representation of the image. Perhaps these inputs could be based on more traditional image features, as described before.

Other applications such as road following (Pomerleau, 1992) and gaze tracking (Baluja & Pomerleau, 1994) have had success with pixel input representations, and one reason for this is that spatial location and variation play an important role in determining the final outcome. In this application, both rotation and spatial invariance must be achieved, which is a much harder task. Additionally, as mentioned before, the range of images which are likely to be encountered in this task is much larger than in other tasks using pixel input representations.

Other factors which make this problem difficult arise not only in the ANN techniques for classification but also in the user's ranking of images. In the set of 400 images originally collected, several images which were exactly the same were ranked differently when given to the user to rank twice, indicating inconsistencies in the user's rankings. To make the problem harder, the user's preferences may be based upon small features in the image. For example, the image in Figure 5(e) would very rarely be picked for recombination by a fully trained ANN, as it is very similar to a simple blank screen.

All of the exploration of architectures and training sets described in the previous sections was done to develop an ANN to simulate a human user's behavior and preferences with respect to a genetic image generation system. In the automated system, the ANN is allowed to direct the genetic search. Several methods of implementing the automation were considered. The first method simply chose the two images to which the ANN gave the highest evaluation to be the parents of the next generation. However, the need for maintaining diversity in subsequent populations weighed heavily against this idea. As discussed previously, GAs need to preserve diversity in order to perform extensive exploration. If only the best candidate chromosomes are selected for recombination, the chromosomes produced rapidly become very homogenous; this can severely limit the potential for further genetic search. In order to avoid stagnation, an alternative method was adopted. In this implementation, the value the *ANN* returned for each image was used as a 'fitness' value. This fitness value was used to determine the image's probability for recombination. Images classified with higher fitness values had a higher probability of being chosen for recombination than those classified with lower ones. However, the best are not guaranteed to be chosen. Standard GAs also use similar probabilistic methods of selection.

Although the probabilistic method yielded good results, the automated system did not as yet incorporate the image repository. In order to provide a small mechanism to escape from very homogenous populations, the automated system selected images from the repository randomly, with a small probability, every generation. The repository was initialized with random images during system start-up. Using the repository allowed for more exploration to be conducted and provided an escape from too homogenous populations. The image repository has not yet been used to store images in the automated system.

The success of this project must finally be judged by the system's ability to create aesthetically pleasing images with minimal interaction of the user. Given this criterion, the results achieved are mixed and very difficult to quantify. The system is able to prune out bad images successfully. However, the judgements on the better images are not as accurate, and some of the better images are also mistakenly pruned away. Throughout this paper, the system has been evaluated on its ability to predict the user's preferences on a static image set. Although this is straightforward, in simulating evolutions, evaluation is much more difficult since the underlying process of evolution is stochastic. The decisions the *ANN* makes are combined with the random factors from the GA before the final product is produced. For example, the evaluations provided by the *ANN* are used as fitness values—they are used to select *probabilistically* the images from the population.

Further, other relevant characteristics of a human user are also hard to implement. In general, users have a very low patience when the current population converges to very similar images, and are apt to restart the program quickly, or to load alternative image libraries. However, the *ANN* has neither a notion of population convergence, nor the ability to reinitialize the population. The *ANN* provides an evaluation for each image based solely upon the image, and not on the population in which the image is contained. These are severe constraints in simulating a user. Nevertheless, in a more limited context, that of providing evaluations of the images to the GA so that it can continue its search process, the user's task can be automated by this project. The goal of this project was to create an end-to-end system which can automatically create pleasing images. Although a few of the automated evolutions have failed, as they converged very quickly and were unable to perform sufficient exploration to find pleasing images, other evolutions, in which convergence occurred much more slowly, revealed complex and interesting images.

An important question, which should be considered along with these results, is how important it is to have a learning agent trying to simulate the user. Would it be possible to get equally good results just using a random selection process? In the experiments attempted in this study, the *ANN,* in comparison to a random process which has a uniform probability of selecting any image, typically produces much more complex images, and maintains diversity in the population longer than simple random selection. One of the reasons that random selection performs poorly is that once a population loses diversity, it is difficult to regain it, given the cross-over and mutation operators used. For example, if two images which are each represented by small equations are recombined, the children produced are likely to be similar. The ANN-based selection method has the advantage of not generally selecting these images, because in its training set such images were usually given low evaluations by the user. As the ANNs used in this study have been given an extensive amount of training data with images which are not considered pleasing, the *ANN* is more likely to avoid selecting images which are represented by smaller equations, and thereby more likely to avoid convergence. Nonetheless, because the *ANN* makes errors in its judgements and because of the stochastic nature of the GA and the small population size used, the populations evolved in this study all eventually converge, whether they are evolved automatically, or manually. The success or failure of a particular evolution must be judged by the amount of useful exploration that is conducted before the population converges.

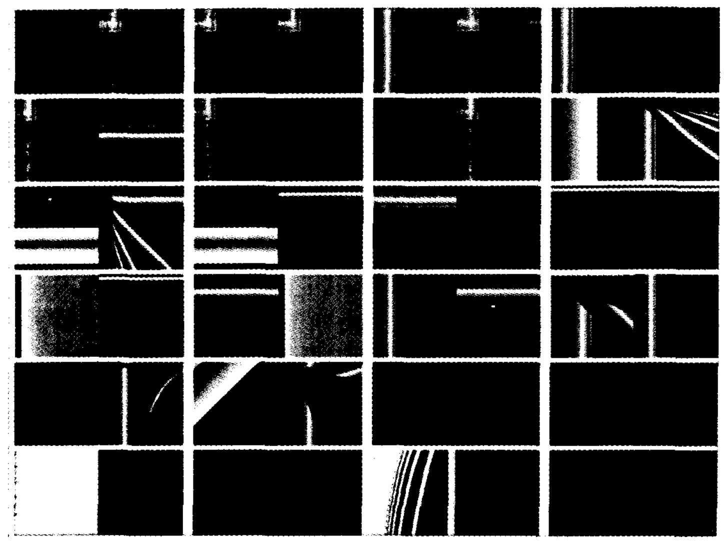
In order to show how the automated evolution typically progresses, Figure 12 presents two evolutions of 24 generations each. The figures show the two parents chosen for recombination in each generation. Figure 12(a) shows a typical evolution using uniform random selection for the images. As can be seen, the exploration is very limited and largely uninteresting. Because of the small probability of choosing randomly from the image repository, sometimes images from outside the population are chosen as parents. However, as the selection process is entirely random, the interesting children produced by the novel parent image may not be chosen again for recombination. Therefore, the population will again quickly converge to uninteresting images. The second set of images, Figure 12(b), shows a typical evolution which is controlled by the *ANN.* Immediately, a vast difference between Figure 12(a) and Figure 12(b) is apparent. The majority of the images chosen are more complex than those which appear in Figure 12(a). However, as can be seen, the network makes mistakes in its selection procedure. Towards the end, these mistakes happen in consecutive generations, the population loses its interesting properties, and quickly becomes too homogenous to perform further interesting search until an image from the repository is randomly selected.

Figure 13 presents 6 images which do not appear in Figure 12. These images were also selected for recombination by the *ANN.* These images were hand-picked from the results of several automated evolution sessions. In comparison to the images shown in Figure 5, which were manually evolved through many hours of interactive time with the system, the images in Figure 13 were selected by the authors in only a few minutes of interactive time with the automated system. One of the potential methods of using this system is to prune away the large number of images which the user will probably dislike, and to present only images which the user is likely to find interesting. Such methods of extending the ability to learn about the user's preferences to work in conjunction with the user is an area for future research. This concept is expanded upon in the next section.

7. Conclusions and Future Directions

The idea of hiding the details of how the mathematical functions are generated, recombined and stored, provides users with the ability to concentrate on the aspect of the algorithm with which they are concerned: the images. By simply selecting images which the users find interesting, it is possible to evolve subsequent populations of images which are closer to individual conceptions of 'interesting'.

Evolution is a powerful method for creating images (Sims, 1991; Todd &



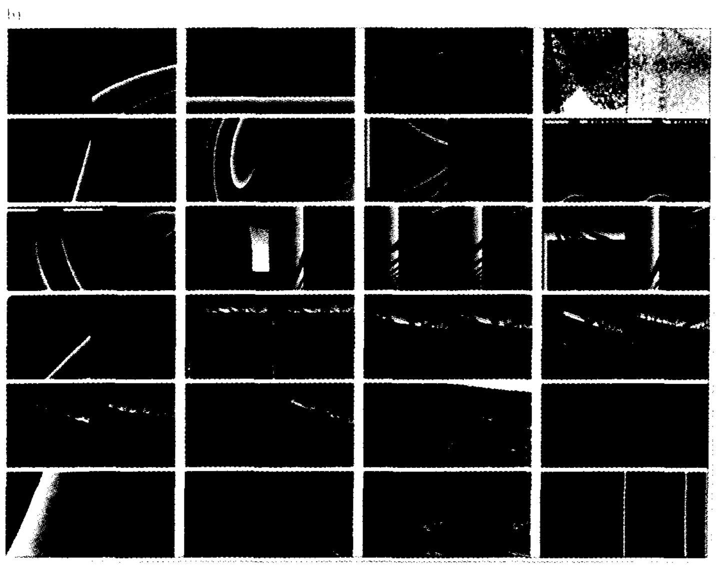


Figure 12. Two automated evolutions. The first is directed by a uniform random selection process. The second is directed by a trained *ANN*. The parents of 24 consecutive generations are shown. The parents of generations **1** are shown in the upper-left comer of the image. The sequence of consecutive generations proceeds from left to right. The lower-right comer images are the parents of generation 24.

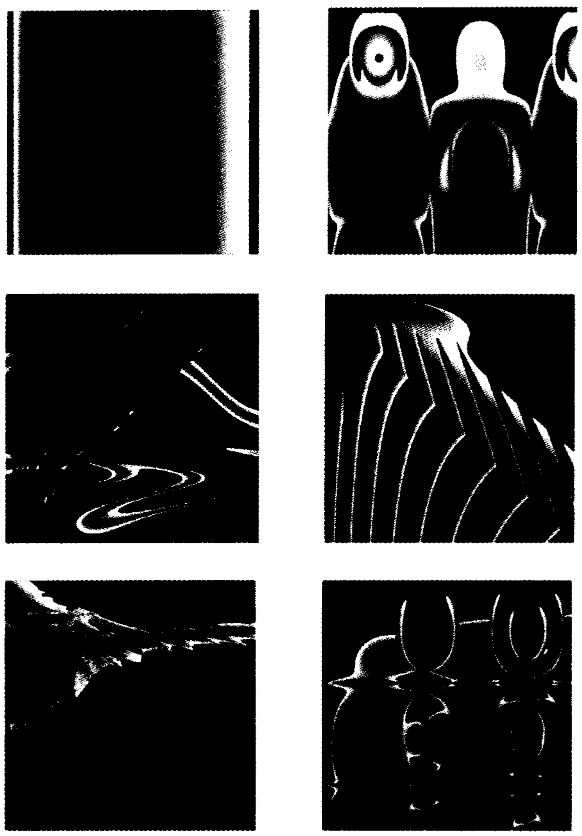


Figure 13. Shown here are six images which the *ANN* selected for recombination through several different evolutions.

Latham, 1992). Although there exists a large random factor in the images which are created, through a number of generations, images will be produced which are closer to the user's desires. Sims stated concisely the role of evolution in this process:

> Evolution is a method for creating and exploring complexity that does not require human understanding of the specific process involved. This process of artificial evolution could be considered as a system for helping the user with creative explorations, or it might be considered as a system which attempts to 'learn' about human aesthetics from the user. In either case, it allows the user and computer to interactively work together in a new way to produce results that neither could easily produce alone. (Sims, 1991).

With the addition of *ANNs*, we hoped to increase the ability for the system to learn and automatically create aesthetically pleasing images which are focused upon the individual tastes of a specific user. Although we cannot claim to have taught the system to understand human aesthetic values, we have made a step in the direction of teaching the system how to simulate an individual user's aesthetic preferences.

Although *ANNs* hold the promise of being able to generalize from examples, the examples must be carefully chosen. To illustrate this point, consider giving only the raw image to the system without the appropriate corrections made for the large color cliff between white and red. Although it is theoretically possible for the ANN to find the distinction, practically, it is helpful to accentuate the differences. This feature, although visually striking, may be lost in other features of the pixel image. Second, consider giving only the sequence of images from a single evolution. Even if the training set contains good and bad images, the images will probably not be heterogeneous enough to allow the *ANN* to generalize to images from another evolutionary sequence. Instead, the ANN must be trained on a variety of images taken from several evolutions.

The GA is a good basis for the system; however, much more can be done to improve the ability of automating the evolutionary process. Further exploration should be divided into at least five areas. The first is that of the neural network architecture. As can be witnessed in the several attempts shown here, different architectures have the potential to lead to very different success rates. However, larger architectures also become increasingly difficult to train, as the need for larger training sets becomes more pronounced. Another architecture which should be considered is an unsupervised learning model in which the similar features are first ascertained, and the images placed into groups based upon similarity. With the formation of these groups, learning the user's rankings of the images may prove to be an easier task.

The second area for possible future attention is creating a fuller, more representative, training set. **As** has been addressed throughout the paper, it is difficult **to** get a large set of images which are in the 'medium' and 'high' range. The training set used in these experiments was derived from several manual evolutions. As the *ANN* is very sensitive to the input data used for training, a larger training set, with many more images classified in the higher regions, may greatly improve performance.

The third area for research requires relaxing the constraint of imbedding no *a* priori knowledge into the *ANN,* and training the system using other characteristics of the image in addition to the pixel image, or as a replacement for the pixel image. These additional features may yield more accurate classifications. Such features might include those commonly used in traditional machine vision techniques; they include clusters, edges, circles, regions of noise, etc. The human eye can easily gauge many of these features automatically and incorporate them into decision-making processes. Giving these features explicitly to an ANN may also prove **to** be beneficial.

The fourth area of future research is examining the effects of using a larger population size. The population size was limited to the nine images in the user interface to make the role of the user easier. However, in an automated system, this is not a concern, and a larger population can be employed. Additionally, more traditional **GA** techniques may be used for generating subsequent populations. For example, in this study, only one set of parents was chosen to create all of the children of the next generation. A more effective means of ensuring diversity is to select different pairs of parents probabilistically to contribute a single 'child' image to the next generation. In this way, multiple parents are allowed to contribute to the subsequent populations. This should have a tremendous beneficial impact in preserving diversity.

Finally, methods for tying the learning mechanisms more closely with the user interface should be explored. One can imagine a system which can work with the user by proposing images which the user may find interesting, while pruning away images which the user will probably dislike. This task would fit well with the abilities of the current system. In addition, the system could also suggest its choice for recombination. If the user disagrees with the system's selection, the user can select an alternative image. If an alternative image is chosen, the learning mechanisms can revise its knowledge to better model the user's preferences. As an extension, perhaps this system can run in the background, periodically querying the user for feedback, and continuing with its defaults when no feedback is given.

Acknowledgements

Profuse thanks are extended to Henry Rowley for his comments and discussions. The authors would also like to thank Kaari Flagstad, Dayne Freitag and Robert Driskill for their aid in revising this paper. Thanks are due to Dr Joseph Bates who introduced us to Karl **Sims'** work, and provided valuable discussions and ideas. The authors would also like to thank the reviewers and editors for providing suggestions and extensions to the system.

Shumeet Baluja is supported by a National Science Foundation Graduate Fellowship. Todd Jochem is partially supported by a ARPA Research Assistantship in High Performance Computing administered by the Institute for Advanced Computer Studies, University of Maryland. This research was supported by the Department of the Navy, Office of Naval Research under Grant No. N00014-93-1-0806.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation, ARPA, or the US government.

References

Baluja, *S.* (1992) ***A Massively Distributed Parallel Genetic Algorithm.*** CMU-CS-92-196R. School of Computer Science, Camegie Mellon University.

Baluja, *S.* & Pomerleau, D.A. (1994) Non-intrusive gaze tracking using artificial neural networks. In J.D. Cowan, **G.** Tesauro & J. Alspector (Eds), ***Advances in Neural Information Processing Systems*** *(NIPS>* ***6.*** San Francisco, CA: Morgan Kaufinann.

Baluja, **S.,** Pomerleau, D.A. & Jochem, T. (1993) ***Simulating a User's Preferences: Towards Automated Artificial Evolution for Computer Generated Images.*** CMU-CS-93-198, School of Computer Science, Camegie Mellon University, 1993.

Cohoon, J.P., Hedge, S.U., Martin, W.N. & Richards, D. (1988) ***Distributed Genetic Algorithms for the Floor Plan Design Problem.*** TR-88-12, School of Engineering and Applied Science, Computer Science Department, University of Virginia.

Goldberg, D.E. (1989) ***Generic Algorithms in Search, Optimization, and Machine Learning.*** Reading, **MA.** Addison-Wesley.

Hem, J., Krogh, A. & Palmer, R. (1991) ***Introduction to the Theory of Neural Computation.*** Reading, **MA:** Addison-Wesley.

Koza, J.R. (1992) ***Genetic Programming: On the Programming of Computers by Means of Natural Selection.*** Cambridge, **MA:** The MIT Press.

LeCun, ***Y., et al.*** (1989) Backpropagation applied to handwritten zip code recognition. ***Neural Computation,*** **1,** 541–551, Massachusetts Institute of Technology.

Pomerleau, D.A. (1992) ***Neural Network Perception for Mobile Robot Guidance.*** PhD Thesis, CMU-CS-92-115. School of Computer Science, Camegie Mellon University.

Sims, K. (1991) Artificial evolution for computer graphics. ***SIGGRAPH '91 Conference Proceedings, 25,*** 319–328.

Todd, P. (1989) A connectionist approach to algorithmic composition. ***Computer Music Journal, 13,*** 27–43.

Todd, **S.** & Latham, W. (1992) ***Evolutionary Art and Computers.*** Academic Press, London.

Waibel, A. *et al.* (1989) Phoneme recognition using time-delay neural networks. In A. Waibel & K.F. Lee (Eds), ***Readings in Speech Recognition,*** pp. 393–404. San Mateo, CA: Morgan Kaufmann.

Appendix: Typical Image Equations

The equations for the images grow large very quickly. Understanding the contributions of each element within the context of the equation rapidly becomes cumbersome. It is also interesting to note that many of the sub-expressions either do nothing, or may easily be replaced by constants. The equations for the images in the bottom row of the 9 squares in Figure 5 are, respectively:

avg(mod(x,x),mul(sin(x),mul(sin(y),add(sin(x),sub(sin(y),mod(sin(y),mul(sub(x,su b(sqr(x),mod(sin(x),mod(sin(x),min(sin(x),sub(y,x)))))),sub(x,add(y,add(sin(x),m od(sin(sin(x)),add(sin(x),sub(sin(y),x)))))))))))))

avg(mod(x,x),mul(sin(x), mul(sin(y),add(sin(x),sub(sin(y), mod(sin(y),mul(sub(x,su b(sqr(x),mod(sin(x),mod(sin(x), min(sin(x),sub(y,x)))))),sub(x,add(y,add(sin(mod(x,x)),mod(sin(x),add(sin(x),sub(sin(y),x))))))))))))

avg(mod(x,x),mul(sin(x),mul(sin(y),add(sin(x),sub(sin(y),mod(sin(y), mul(sub(x,su b(hsn(x),mod(sin(x), x))),sub(x,add(y,add(sin(x), mod(sin(x),add(sin(x),sub(sin(y), x)))))))))))))