AN EVOLUTIONARY APPROACH TO FUZZY IMAGE COMPRESSION

Mauro L. Beretta, Andrea G.B. Tettamanzi
Università degli Studi di Milano, Dipartimento di Scienze dell'Informazione
Via Comelico 39/41, I-20135 Milano, Italy

An evolutionary algorithm for the optimization of combined fuzzy compression and decompression of images is introduced. An image is divided into squares, and a fuzzy compression algorithm decides the color of the pixels that will replace the whole squares in the compressed image. A fuzzy decompression algorithm reconstructs the colors of pixels in the decompressed image on the basis of the pixels in the compressed image. An evolutionary algorithm is used to optimize the parameters involved in both fuzzy algorithms. Such an evolutionary algorithm has been implemented on a Quadrics super-computer. The paper provides a brief discussion of experimental results.

1 Introduction

The use of computer networks for video-conferencing and multi-media applications makes the research on fast, efficient image compression algorithms an issue of paramount importance.

When efficiency is more important than precision and detail, fuzzy techniques[6] can play a substantial role.

One approach to (lossy) image compression consists in reducing the number of pixels in a picture by replacing a whole square with a single pixel in a compressed picture.

A premise of this approach is that the integrating power of the human eye is somehow capable of recovering the missing information in such a compressed picture.

The authors have attempted to achieve similar results by combining one fuzzy algorithm to determine the contents of the replacement pixels in a compressed picture with a second fuzzy algorithm to reconstruct a plausible value for the pixels in the final decompressed image.

2 Compression

In the compression stage, the value of the pixel which will replace a square portion of the original image is computed based on the color values of pixels belonging in the square.

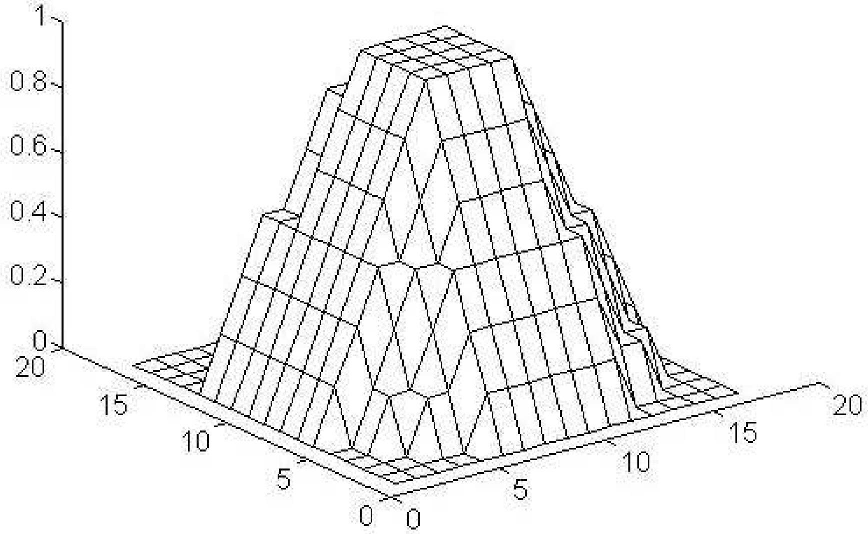

Figure 1: An example of a compression membership function.

This is where fuzziness comes into play: In this fuzzy approach, a "square" is in fact a fuzzy set of pixels defined through an appropriate membership function which tells, for each pixel in the picture, to what degree it belongs in the square.

Such a function assigns nonzero membership degrees also to pixels that would not belong in the (crisp) square. In this way a single pixel is an element (in the fuzzy sense) of more than one square and, as a consequence, the various portions thus defined partially overlap.

This observation is the crux of the whole compression and decompression process, as illustrated in Section 3.

The content of a square must be mapped into a single pixel $\mathbf{x}$ in the compressed picture. The transformation is of the form

$$\begin{aligned} x_C &= w\frac{\sum_i \sum_j \mu_{\mathbf{x}}(i,j) I^C_{ij}}{\sum_i \sum_j \mu_{\mathbf{x}}(i,j)} + \\ &+ (1-w)\sup_{i,j}\{\mu_{\mathbf{x}}(i,j) I^C_{ij}\}, \end{aligned} \tag{1}$$

for each component C, where I is the source image and $\mu(\cdot)$ is a membership

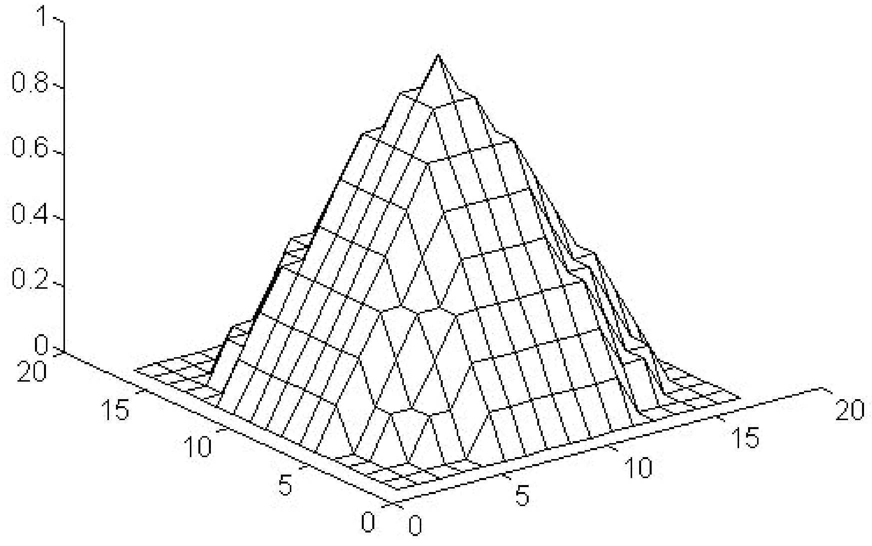

Figure 2: An example of a decompression membership function.

function defining the square, such as the one shown in Fig. 1.

In words, the color for the compressed square is obtained as a convex linear combination of the mean and the mode of the color distribution weighted by the degree of membership in the fuzzy square.

The use of the mode is motivated by the need to preserve true colors of details through the compression/decompression process, since the mean alone tends to assign an average color.

For instance, consider a picture of a face: shouldn't the mode be used, some details, like the white around the eyeballs, would result as grey in the compressed image, due to the averaging effect of the mean between the white color and the other surrounding colors of the face.

3 Decompression

In the decompression stage, the procedure described above is reversed: the value of a pixel in the final image is computed according to the values of the replacement pixel relevant to the squares it belongs in, again with an appropriate membership function.

In general a pixel of the final image will belong in more than one squares, with distinct degrees of membership depending on its position.

Given a compressed image X, obtained as a result of the fuzzy compression algorithm, that is a collection of color values for each square in the source image, the color of a pixel J_{ij} is given by the following expression, for each of the three color components C:

$$J_{ij}^C = \sum_{\mathbf{x} \in X} \pi_{\mathbf{x}}(i,j) x_C, \tag{2}$$

where J is the final, decompressed image and $\pi_{\mathbf{x}}$ (which in general will be distinct from $\mu_{\mathbf{x}}$) is a membership function associated with square $\mathbf{x}$, such as the one shown in Fig. 2.

4 Synthesis and optimization

It is evident that the success of this approach critically depends on the proper definition of many parameters, defining the shapes of the membership functions utilized in both the compression and decompression phase.

At first, an attempt was made at optimizing these parameters by hand, but quite soon it became evident that there were too many alternatives to be explored.

As a consequence, it was decided to resort to an evolutionary algorithm along with the power of a parallel machine, the Quadrics.

4.1 Evolutionary Algorithms

The name *Evolutionary Algorithm* (EA) encompasses a family of stochastic optimization techniques based on the key concept of *evolution,* such as Genetic Algorithms,[3] Evolution Strategies, Evolutionary Programming and Genetic Programming.[4] A recent book of reference and synthesis in that field is.[5]

An EA makes a population of candidate solutions for the problem at hand evolve by iteratively applying a set of stochastic operators, known as *mutation*, *recombination*, *reproduction* and *selection*.

Mutation randomly perturbs a candidate solution; recombination decomposes two distinct solutions and then randomly mixes their parts to form a novel solution; reproduction replicates the most successful solutions found in a population; selection purges poor solutions from a population.

Figure 3: The original picture.

The resulting process tends to find globally optimal solutions to the problem much in the same way as natural populations of organisms adapt to their surrounding environment.

4.2 The Quadrics

The Quadrics parallel supercomputer[1] is a single instruction multiple data parallel hardware, consisting of an integer CPU, which controls the instruction flow and integer arithmetics, and an array of custom floating-point units, each one equipped with its own memory.

The peak computing power delivered by a Quadrics can range from 400 MFlops to 100 GFlops depending on the number of floating-point units installed.

This figure may vary from a minimum of eight floating-point units to a maximum of 128.

4.3 Algorithm Implementation

The parallelism of the Quadrics is taken advantage of both in the evolutionary part of the algorithm and to compute the fitness of the evolved compression/decompression strategies.

The programming language used is TAO, but some bottleneck routines were written in Zz^2, a sort of assembler for that machine.

Encoding A compression/decompression strategy is encoded as an individual made up of 513 floating-point numbers in $[0, 1]$, whose first 256 floating-point numbers parameterize the membership function μ used for compression; the next 256 floating-point numbers parameterize the membership function π used for decompression, and the remaining one is the parameter w controlling the linear combination of mean and mode.

Other encodings might be taken into account in order to reduce the degrees of freedom in the optimization process, for instance taking advantage of symmetry assumptions, as noted in Section 5.

Operators Mutation acts by replacing, with probability 0.0005, each floating point number of the encoded individual with a new floating point number extracted from the interval $[0, 1]$ with uniform probability.

Given this mutation rate, the probability that at least one mutation happens in an individual is 0.226.

The crossover employed is a single-point crossover, with rate 0.6, operating on floating-point numbers. This means that no floating-point number can be split in two pieces by the crossover point.

Fitness A compression/decompression strategy to be evaluated is applied to an entire gallery of images, to compress and decompress them.

The fitness measure is thus a function of the average mean square error between the original and the final images: For each color component C, the error associated with each image pair (I, J),

$$e^C_{(I,J)} = \sqrt{\frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (J^C_{ij} - I^C_{ij})^2}, \tag{3}$$

is computed, where M and N are the sizes of the image, and an aggregated error is obtained as the average of the maximum component-wise errors for each image over the gallery.

Figure 4: The compressed picture.

The fitness is then obtained as the negative exponential of the aggregated error.

Although it doesn't fully capture the perceived, subjective goodness of an approximation, the error function employed is nevertheless the one that proved best in our experiments.

5 Experiments and Results

Experiments using the evolutionary algorithm described above have just begun, and we refer the interested reader to forthcoming articles for a complete discussion of the results.

In a preliminary phase we have carried out several experiments using different compression/decompression strategies, tuning the shapes of membership functions and the mean versus mode combination.

The results have been encouraging, but it soon became evident that the space of possible combinations to explore was too large and that an automatic search procedure, such as the evolutionary algorithm described above, was required.

Figs. 3–5 show three stages in the working of the compression/decompression fuzzy algorithm using a hand-crafted strategy, in order to illustrate the kind of results obtained.

Fig. 3 is the original picture; Fig. 4 is the result of applying the fuzzy compression algorithm with the membership function that appears in Fig. 1. The corresponding compression ratio is 64 to 1.

Fig. 5 is the final result, after applying the fuzzy decompression algorithm with the membership function shown in Fig. 2.

The evolutionary algorithm described above has been implemented on a Quadrics machine with 32 floating-point units. The use of that configuration was dictated by the high demand of the algorithm for computing resources.

The gallery of images used to compute fitness comprises twenty-four close-up portraits of people of both sexes and different ages.

When the form of compression and decompression membership functions is completely unrestricted, convergence of the evolutionary algorithm is rather slow, in the order of hundreds of generations.

To achieve better performance, one could make the assumption that any compression or decompression membership function has to be symmetrical with respect to the center of a square. Finding a shape for such a membership function boils down to determining its shape along the radius departing from the center of the square.

This device would dramatically decrease the number of degrees of freedom and, therefore, it would speed up convergence of the algorithm.

Acknowledgments

The authors would like to thank Prof. Gianni Degli Antoni of the Dipartimento di Scienze dell'Informazione , University of Milan, Dr. Rinaldo Poluzzi of SGS-Thomson for their suggestions and comments and Dr G. Rizzotto of SGS-Thomson for his encouragement and for supporting the collaboration and research.

The authors are also grateful to the staff of INFN, Rome, namely Dr. Federico Massaioli and Dr. Mario Rosati, for support and help in using their Quadrics machines.

References

1. Alenia Spazio S.p.A. *Quadrics Primer*, rev. 1.1 edition, September 1994.

Figure 5: The decompressed picture.

2. S. Cabasino, G. M. Todesco, and P. S. Paolucci. *Zz: The TAO engine.* Alenia Spazio S.p.A., vers. 2.0 edition, March 1995.
3. D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, Reading, MA, 1989.
4. J. R. Koza. *Genetic Programming: on the programming of computers by means of natural selection.* The MIT Press, Cambridge, Massachussets, 1993.
5. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin, 1992.
6. L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.