Obtaining the phase of an interferogram by use of an evolution strategy: Part I.

Sergio Vázquez-Montiel, Juan Jaime Sánchez-Escobar, and Olac Fuentes

We present a method for obtaining the phase of a noisy simulated interferogram. We find the wave-front aberrations by transforming the problem of fitting a polynomial into an optimization problem, which is then solved using an evolutionary algorithm. Our experimental results show that our method yields a more accurate solution than other methods commonly used to solve this problem. © 2002 Optical Society of America

OCIS codes: 100.2650, 220.4840, 220.1010, 120.3180, 120.5050.

1. Introduction

Fitting a polynomial to the optical path difference (OPD) for interferometric patterns is a difficult task. Dutton *et al.*[1] were the first to report that when the least-squares method is used to find the fitting polynomial, the results obtained are commonly inconclusive. The same results were reported even when the polynomial was orthogonal over the data points.[2] When noise is present, the least-squares method often leads to overfitting, as the rms error keeps decreasing when more higher-order terms are included in the polynomial.[3]

Analyzing a noisy interferogram with the rms criterion, it can select randomly the number of terms in the polynomial, which increases overfitting when the noise level increases and when the number of fringes decreases.[4] Thus the rms criterion does not lead to a conclusive selection of the polynomial terms, even for the simplest interferogram affected by tilt only.[5] However, in a recent article[6] an algorithm that finds the polynomial order satisfactorily was shown.

We propose an algorithm to find the phase of a noisy simulated interferogram by transforming the problem of fitting a polynomial into an optimization problem in real domains, which is then solved using the evolution strategy, a nontraditional optimization algorithm based on the mechanisms of biological evolution. In Section 2 we describe the procedure to generate a noisy simulated interferogram. Section 3 presents a generalized description of the evolutionary algorithm and the evolution strategy. Section 4 gives a detailed description of our algorithm. In Section 5 the main results are shown, and Section 6 presents conclusions.

The authors are with the Instituto Nacional de Astrofísica, Optica y Electrónica, Apdo. Postal 216, Puebla 72000, Pue, México. S. Vázquez-Monteil's e-mail address is svazquez@inaoep.mx.

Received 20 August 2001; revised manuscript received 16 January 2002.
0003-6935/02/173448-05$15.00/0
© 2002 Optical Society of America

2. Noisy Simulated Interferogram

To obtain the noisy simulated interferograms we use Kingslake's formulation, where the irradiance in the interferogram plane is given by

$$I(x, y) = \left\{A + B \cos\left[\frac{2\pi}{\lambda} W(x, y)\right]\right\} + \text{Gaussian error.} \quad (1)$$

The Gaussian error is simulated with two properties, namely, the mean and the standard deviation. Over each fringe the mean is zero and the standard deviation is σ_m. λ is the wavelength and $W(x, y)$ is the OPD, and it is represented by a polynomial using the Seidel aberration

$$W(x, y) = A(x^2 + y^2)^2 + By(x^2 + y^2) + C(x^2 + 3y^2) + D(x^2 + y^2) + Ey + Fx, \quad (2)$$

where A is a spherical aberration coefficient, B is a coma coefficient, C is a astigmatism coefficient, D is a defocusing coefficient, E is a tilt about the y axis, and F is a tilt about the x axis.

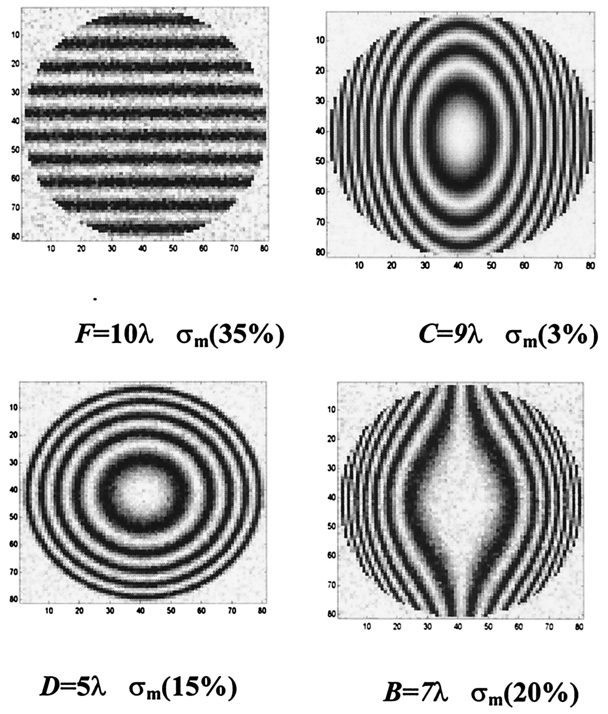

Fig. 1. Noisy simulated interferograms.

Some examples of noisy simulated interferograms are shown in Fig. 1.

3. Evolutionary Algorithms

Evolutionary algorithms are a family of learning and optimization methods, inspired in the notion of biological evolution by means of natural selection. They are relatively insensitive to noise and immune to local minima, which makes them well suited for solving optimization problems in complex or poorly modeled domains.

An evolutionary algorithm works with a set of potential solutions to the problem of interest, where each potential solution, called an individual, is represented by a string of characters. The main objective is finding a string of characters that encodes an adequate solution to the problem.

The evolutionary algorithm starts searching for a solution within a population of individuals, each of which represents not only a search point in the space of potential solutions to a given problem, but also may be a temporal container of current knowledge about the laws of the environment. The starting population is created by an algorithm-dependent method, and evolves toward successively better regions of the search space by means of randomized processes of recombination, mutation, and selection.

The environment delivers a quality metric (fitness value) for each search point, and the selection process favors those individuals with higher quality to reproduce more often than weak individuals.[7]

The recombination mechanism allows for mixing of parental information while passing it to their descendants, and mutation introduces innovation into the population. This process is used in the three different types of evolutionary algorithms, i.e., genetic algorithms (GAs), genetic programming (GP), and evolution strategies (ESs),

In genetic algorithms, individuals are represented by fixed-length strings over a small alphabet (usually binary). Their main application lies in solving discrete optimization problems where conventional algorithms cannot find solutions in reasonable amounts of time.[8] Of particular interest is the class of NP-complete problems (Cook, 1976), a large class of binary optimization problems for which no efficient algorithms have been found.

Individuals in genetic programming are encoded as trees, and are interpreted as programs written in a programming language similar to the LISP.[9] GP has been used to solve some difficult planning and design problems.

ES are a class of optimization algorithms that find near-optimal solutions to ill-defined problems applying operators inspired in the mechanics of biological evolution, including crossover, mutation, and the survival of the fittest. In its simplest form, optimization starts with a set of n parents, where each parent represents a real-valued vector that encodes a candidate solution to the problem at hand.[10] The following two steps are then repeated until a termination condition is attained: (1) Create k offspring (where $k < n$), applying recombination to a fraction of the parents (crossover), and create $n - k$ offspring applying random changes to the rest of the parents (mutation). (2) From the resulting $2n$ individuals, choose the best n as the parents for the next iteration (selection). The process terminates when a prespecified number of iterations or generations are executed, or a goal value in the objective function is attained, for example, in this article the objective function is based into Euclidian space and the goal value is found when we obtain the distance minimum between two images. Crossover is realized by combining elements from the parents of an individual. According to the biological observation that offspring are similar to their parents and that small changes occur more often than large ones, mutation is realized by adding to the parents a normally distributed random vector with an expected value of zero.

A major feature of this family of algorithms is the dynamic updating of the standard deviation of the distribution used to obtain the descendant in response to the characteristic of the region of the objective function that is being explored. If successful mutation occurs rarely, the search is likely to be near a minimum, and thus it is a good idea to decrease the size of the neighborhood being searched. If successful mutations occur very often, it means that convergence could be sped-up by increasing the step size.

The evolution strategy has been shown to be globally convergent given unbounded running time.[11] Similar results have also been shown for simulated annealing[12] and genetic algorithms.[13] Of more practical interest, the evolution strategy has been shown in many applications to converge quickly and be relatively insensitive to local minima. The evolution strategy is generally preferable to genetic algorithms for solving problems that deal with the optimization of functions of real numbers. Its main advantage over simulated annealing lies in its adaptive step length, realized by dynamically varying the standard deviation of the mutation vector in response to the characteristic of the objective function and the region being explored.

4. Optimization Algorithm

The optimization algorithm takes as input a noisy simulated reference interferogram and gives as output the vector of aberration coefficients that best matches it. Its operation is explained by means of the following pseudocode:

```
$t := 0;$
    Initial Population $P_{\text{parents}}(0) = \{\boldsymbol{a}_1(0), \ldots, \boldsymbol{a}_\mu(0)\}$,
        where $\boldsymbol{a}_k = (x_{k,1}, \ldots, x_{k,i}, \sigma_{k,1}, \ldots, \sigma_{k,i}),\ k \in \{1, \ldots, \mu\}$.
Evaluate fitness of $P_{\text{parents}}(0)$: $\{f[\boldsymbol{a}_1(0)], \ldots, f[\boldsymbol{a}_\mu(0)]\}$.
While (Termination condition $\neq$ true) do
        $P_{\text{offspring}}(t) =$ Generate $(\mu * ra)$ offspring using recombination,
            $\{\boldsymbol{b}_1(t), \ldots, \boldsymbol{b}_{\mu * ra}(t)\}$.
            Generate $[\mu * (1 - ra)]$ offspring using mutation,
            $\{\boldsymbol{b}_{\mu * ra} + 1(t), \ldots, \boldsymbol{b}_\mu(t)\}$.
Where $(ra)$ is a constant in the [0,1] range.
Evaluate fitness of $P_{\text{offspring}}(t)$: $\{f[\boldsymbol{b}_1(t)], \ldots, f[b_\mu(t)]\}$.
Let $P_{\text{parents}}(t+1)$ be the $\mu$ individuals from $P_{\text{offspring}}(t)\ \mathrm{Y} P_{\text{parents}}(t)$ with the highest fitness
$t = t+1;$
    end.
```

The fitness of an individual, $a_k = < x_{k,1}, x_{k,2}, x_{k,3}, x_{k,4}, x_{k,5}, x_{k,6}, \sigma_{k,1}, \sigma_{k,2}, \sigma_{k,3}, \sigma_{k,4}, \sigma_{k,5}, \sigma_{k,6} >$, is evaluated as follows:

a) Construct interferogram G_k from a_k, by substituting $A = x_{k,1}, B = x_{k,2}, C = x_{k,3}, D = x_{k,4}, E = x_{k,5}$, and $F = x_{k,6}$ into Eq. (2).
b) Considering each pixel in an interferogram as a dimension in a Euclidean space, assign as the fitness of I_k the negative of the distance from G_k to the reference interferogram I.

$$\text{Distance}\,(G_K, I) = \left[\sum_{i=1}^{m}\sum_{j=1}^{n}(G_{Ki,j} - I_{i,j})\right]^2. \qquad (3)$$

For the recombination operator, we use the following two alternatives: Local intermediary recombination, which works by selecting randomly the parents for each vector component and computing a weighted sum of the corresponding components of the parents

$$\boldsymbol{b}_{rec1k} = r_k\, \boldsymbol{a}_{\rho 1,K} + (1 - r_k)\boldsymbol{a}_{\rho 2,k}, \qquad (4)$$

where $r_k \sim \mathrm{Y}([0, 1])$ or $r_k = 1/2$ and $(\rho 1, \rho 2) \sim \mathrm{Y}(\{1, \ldots, \mu\})$ for each offspring. Discrete recombination, where each vector component is copied from the corresponding component of an individual randomly chosen from the parents

$$\boldsymbol{b}_{rec2k} = \boldsymbol{a}_{\rho k,k}, \qquad (5)$$

where $\rho_k \sim \mathrm{Y}(\{1, \ldots, \mu\})$ at random for each k.

The mutation operation consists of making random changes, adding to each object variable $x_{k,i}$ a random variable taken from a normal distribution with mean equal to zero and deviation standard equal to $\sigma_{k,i}$, that is

$$\boldsymbol{b}_{mut\,k} = [x_{k,1} + M(0, \sigma_{k,1}), \\ x_{k,2} + M(0, \sigma_{k,2}), \ldots, \\ x_{k,i} + M(0, \sigma_{k,i})]. \qquad (6)$$

The σ strategy parameters are updated using the 1/5 rule. If the estimated probability of successful mutation is greater than 1/5, σ is increased, otherwise it is decreased, that is

If $P(f(b_{mut\,k}) > f(b_k)) > 1/5$

$$\sigma_{k,i} = \sigma_{k,i} * (1 + c),\ i = 1, \ldots, n,$$

else

$$\sigma_{k,i} = \sigma_{k,i}/c,\ i = 1, \ldots, n,$$

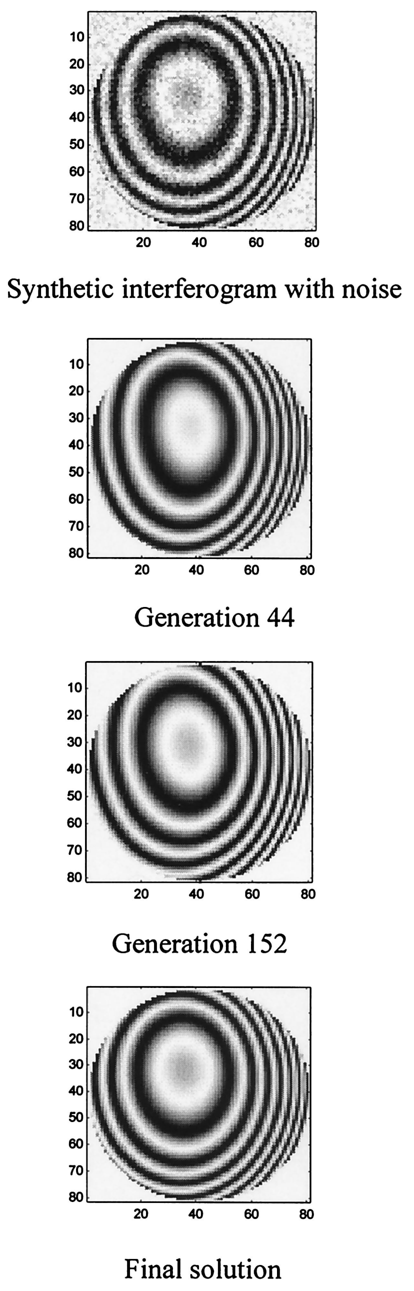

Fig. 2. Interferograms obtained in different generations within the optimization process.

where c is a small positive constant. For the experiments we used $c = 0.01$.

5. Computer Simulation Results

The algorithm described in the previous section was implemented in the Matlab™ programming language. In all of our experiments we used a population size $\mu = 50$, we applied local recombination Eq. (4) to obtain 10% of the offspring, discrete recombination Eq. (5) to obtain another 30%, and mutation Eq. (6) to obtain the remaining 60% of the offspring.

We evaluated the fitness of each new individual as described previously, and selected the best μ individuals from the set containing both the parents and the offspring as the parents for the next generation. As a termination condition we used a fixed number of generations, 1000 in this case. After the program is run for this number of generations it outputs as a solution the object variables of the best individual found in the last generation. Each experiment took approximately 10 min to run on a Pentium III computer.

Figure 2 shows an optimization process. The algorithm analyzed an interferogram that contains: A, B, C, D, E, F equal to 1λ, and a noise level σ_m of 35%. Table 1 shows the results, verifying that our program works satisfactorily using all the coefficients. Figure 3 shows the fitness of the best individual plotted against the generation number in a typical run of the algorithm. It can be seen that most of the progress occurs early in the search, and that the number of generation and thus the running time can be decreased by half without loss in terms of the accuracy of the solution found. This type of behavior is very common in stochastic optimization algorithms.

Table 1. Comparison between the Initial Aberrations and the Results of the Algorithm

Analyzed Interferogram	Solution	Error
$A = 1\lambda$	$A = 0.9976\lambda$	0.0024λ
$B = 1\lambda$	$B = 1.0012\lambda$	0.0012λ
$C = 1\lambda$	$C = 1.0014\lambda$	0.0014λ
$D = 1\lambda$	$D = 0.9976\lambda$	0.0024λ
$E = 1\lambda$	$E = 1.0045\lambda$	0.0045λ
$F = 1\lambda$	$F = 0.9994\lambda$	0.0006λ

6. Conclusions

In this work we showed a method to obtain the phase of a noisy simulated interferogram based on evolutionary computation. The algorithm has several advantages over traditional methods: The algorithm finds not only the right coefficients, but also the num-

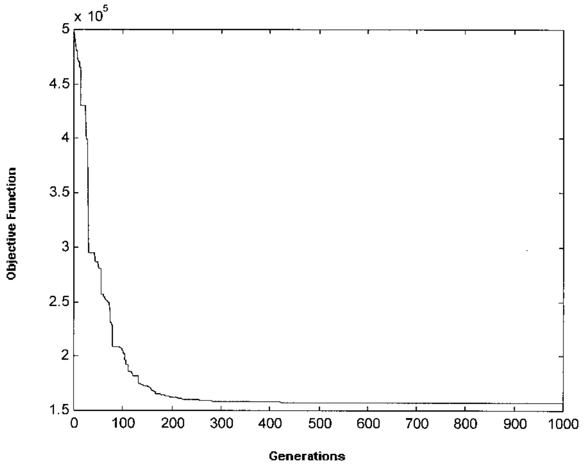

Fig. 3. Behavior of the objective function. The evolution strategy converges to the solution in generation 430.

ber of terms of the polynomial used in the construction of the synthetic interferogram. It is not necessary for the user to introduce additional data, only those related with the aberration coefficients to generate the interferogram that is analyzed. Also, our algorithm fits a polynomial in an automatic form, even when a considerable noise level, $\sigma_m = 35\%$, affects the interferogram.

In the case of an analyzed interferogram with equally spaced straight fringes parallel to the *x* axis (two wavelengths) and noise (35%), we obtained accurate coefficient values. (The comatic term was nearly equal to zero). We obtained similarly good results when Seidel aberrations are included, and when Seidel aberrations and tilts are combined.

Our program introduces two important characteristics:

- High precision with the values of the coefficients that are found.
- Fitting to the polynomial in an automatic form, without overfitting.

7. Future Work

Future work includes making a determination of the phase of experimental interferograms. Initially we will be introducing a vision technique for obtaining an enhanced contrast in respect to the interferogram analyzed. Our method will be compared with the least squares fit method.

References

1. D. Dutton, A. Cornejo, and M. Latta, "A semiautomatic method for interpreting shearing interferograms," Appl. Opt. **7,** 125–131 (1968).
2. G. E. Forsythe, "Generation and use of orthogonal polynomials for data-fitting on a digital computer," J. Soc. Ind. Appl. Math. **5,** 74–80 (1957).
3. J. Y. Wang and D. E. Silva, "Wave-front interpretation with Zernike Polynomials," Appl. Opt. **19,** 1510–1518 (1980).
4. A. Cordero-Dávila, A. Cornejo-Rodríguez, and O. Cardona-Nuñez, "Polynomial fitting of interferograms with Gaussian errors on fringe coordinates. I: Computer simulations," Appl. Opt. **33,** 7343–7349 (1994).
5. A. Cordero-Dávila, A. Cornejo-Rodríguez, and O. Cardona-Nuñez, "Polynomial fitting of interferograms with Gaussian errors on fringe coordinates. II: Analytical study," Appl. Opt. **33,** 7343–7349 (1994).
6. A. Cordero-Dávila, A. Cornejo-Rodríguez, and O. Cardona-Nuñez, "Polynomial fitting of interferograms with Gaussian errors on fringe coordinates. III: Nonlinear solution," Appl. Opt. **37,** 7983–7987 (1998).
7. T. Back, *Evolutionary Algorithms in Theory and Practice,* (Oxford University Press, New York, 1996).
8. D. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning,* (Addison-Wesley, Reading, Mass. 1989).
9. J. Koza, *Genetic Programming: on the Programming of Computers by Means of Natural Selection;* (MIT Press, Cambridge, Mass., 1992).
10. O. Fuentes, R. C. Nelson, "Learning dextrous manipulation skills for multifingered robot hands using the evolution strategy," Machine Learning, **31,** 223–237 (1998).
11. J. Born, "Evolutionstrategien zur numerischen Losung von Adaptionsaufgaben" Ph.D. dissertation Humboldt Universitat, Berlin, Germany, 1978).
12. E. H. L. Aarts, and J. Korst, *Simulated Annealing and Bolzmann Machines,* (Wiley, Chichester, 1989).
13. A.E. Eiben, E. H. L. Aarts, & K. M. Van Hee, "Global convergence of genetic algorithms: an infinite Markov chain analysis," *Proceeding of the First International Conference on Parallel Solving from Nature*, (Springer, Berlin, Germany, 1991) pp. 4–17.