

Evolutionary Algorithm for 3D Object Reconstruction from Images

Renata L. M. E. do Rego

rlmer@cin.ufpe.br

Hansenclever F. Bassani

hfb@cin.ufpe.br

Aluizio F. R. Araújo*

aluizioa@cin.ufpe.br

*Federal University of Pernambuco,
Center of Informatics, Av. Prof. Luis Freire s/n,
Cid.Universitária 50732 - 970, Recife - PE –
Brazil*

Fernando Buarque de Lima Neto

fbln@dsc.upe.br

*University of Pernambuco – Polytechnic School
of Engineering, Department of Computing
Systems, Rua Benfica, 455 – Benfica, 50.720-001,
Recife - PE - Brazil*

** Corresponding author*

Abstract

This work presents an evolutionary algorithm to reconstruct 3D objects based on images of them. In the proposed evolutionary algorithm, we describe a way to evolve 3D initial models to a target object by means of comparisons between images generated from the models and images of target object, in which the acquisition position is known. The proposed a modification in the standard evolutionary strategy algorithm that produced better and faster results in the class of problems at hand (i.e. when many genes must evolve mainly to a direction in the search space). Satisfactory results were achieved reconstructing simple 3D objects, such as an ellipsoid object and a pear from a sphere. Although performance decrease for more detailed objects such as a face, the proposed solution still converges.

1. Introduction

In the past few years, great advances were achieved in technologies for visualization of 3D models [1] allowing the use of personal computers to process and to display 3D complex scenes in real time. Also, there is a growing need for automatic techniques that produce 3D models such as real world objects representations. For example: in Medicine, models of body parts are used for planning surgeries and academic training [2]; in Architecture, toy models help visualization of buildings; and in entertainment, the game industry (and their users) benefits greatly from the “realism” of artificially generated models. These models can be designed by labor-intensive CAD

processes or obtained by expensive reverse engineering processes that are based on recent scanning technologies [1].

This work introduces an evolutionary approach for 3D object reconstruction based on images taken from it. The evolutionary approach was successfully applied in solving problems, which are similar to the one at hand and is used when: (i) inputs and outputs are known; (ii) the solution path is not known or well structured; and (iii) only methods for evaluating candidate solutions are known (e.g. Ackley function) [6].

Evolutionary Strategies (ES) is the chosen approach because they are thought to be more suitable for the present problem. The standard ES of Eiben and Smith [6] was the start point because representations are floating-point vector, suitable to represent vertex coordinates; the variation operators are discrete or intermediary recombination and self-adaptive mutation is quite appropriate to the current problem. Typically, the self-adaptive mutation operator allows various magnitudes of mutation in different stages of the evolutionary process.

The evolutionary process searches for a solution from an initial object, a model loosely related to the target, through successive evolutionary steps. Evolution works analogously to other approaches for reconstruction task based on deformable objects such as: irregular meshes [3], finite element [4] and balloon models [5]. Even though, the proposed algorithm differs from previous algorithms in two ways: first, “position searches” are transformed into “direction searches” to change paradigm from “finding a better solution.” to “finding a better way to find better solutions”. Secondly, selection is rank-based instead of uniformly random to stress depth search.

Different representations of 3D models are presented by Campbell and Flynn [1]. *Polygonal Meshes* represents an object as a list of vertices and polygons. These representations allow objects to be easily deformed by modifying vertices positions.

In this work, the system inputs are points-of-view of the real world object (*i.e.* image and the position of its acquisition) and one initial model. The system outputs are 3D models of the object being modeled. The evolutionary algorithm searches for models that best resemble the original object by generating, combining and testing many candidate models. The quality of the solution is assessed by comparing differences between images of the generated models and images of the actual object.

A major condition to produce a model with irrelevant differences between input images and model images is to make use of same conditions of illumination and object colors. Because generating such environment is not the main objective of this work, we used as target, objects made by of-the-shelf modeling tools [7]. Then, from now on, we refer to the real object, as target object. This approach follows another important constraint: the connectivity between vertices of any evolving model should match the connectivity of the initial mesh.

Simulations compared the efficiency and the efficacy of the proposed algorithm, and the standard version of Evolutionary Strategies [6] (Section 3). The results of the experiments suggest that performance is related with the number of vertices needed to model the object. Hence, our algorithm performed satisfactorily in simple objects such as sphere-, egg-, pear- and apple-shaped, as opposed to poorer results in more complex objects, such as faces.

Section 2 includes all elements of the proposed solution, with especial focus on the evolutionary algorithm put forward. The simulation results of are presented in Section 3. This paper conclusion, including some discussion and future work, is in Section 4.

2. Proposed system

This section describes the proposed system to generate 3D models using an evolutionary approach. This system is called Evolutionary Reconstruction System (ERS).

2.1. Input and output of the ERS

The ERS receives, as input, points-of-view (*i.e.* the pair image and coordinates), which are pairs of (i) actual gray scale image of the object and (ii) Cartesian (x, y, z) coordinates representing the point in space

where the images were acquired from. It was assumed that the object is at the origin of the Cartesian system, the camera points towards it (coordinates: 0, 0, 0), and has zero as rotation angle in relation to x-axis, *i.e.* twist. For simplicity sake, all images are in the gray scale color format even though they might be extended to RGB easily. The background color of the images must be marked with a special color not used inside the object. A suitable selection of images is crucial for reconstruction.

At the end of computations (*i.e.* all evolutionary steps) the aim of the system is to produce a model that precisely represents the input object (*i.e.* target object).

2.2. Architecture of the ERS

The system components (Figure 1) are: (i) 3D engine, (ii) image comparator and (iii) evolutionary algorithm. Items (i) and (ii) are described in this subsection and item (iii) is detailed in the next subsection.

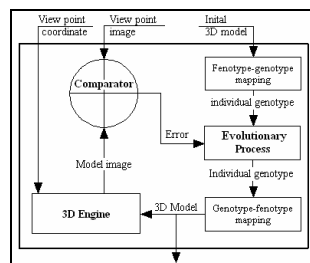


Figure 1 – Schematic view of the ERS architecture.

Image Comparator

The image comparator receives, as input, two images and yields a value representing the difference between them. The comparison of images is carried out pixel-by-pixel as described in Equation (1):

$$\text{difference}(img_1, img_2) = \sum_{i=0}^w \sum_{j=0}^h (\text{pixdif}(img_1.\text{pixel}(i, j), img_2.\text{pixel}(i, j)))^2 \quad (1)$$

where, w is width, h is height of images and

$$\text{pixdif}(pix_1, pix_2) = \begin{cases} \text{maxvalue, if one pixel is} \\ \text{background and other is not.} \\ |pix_1 - pix_2|, \text{ otherwise} \end{cases} \quad (2)$$

where maxvalue is the maximum of: $2 * |pix_1 - pix_2|$.

The pixdif function either penalizes models that do not cover the entire image of the target object and the models covering areas outside the target object image or the function penalizes softly models that have

different gray scales inside the target object image area.

3D Engine

The 3D engine is responsible for generating new 3D images based on the evolving models. Each image of a model is produced from the same position where the original one was acquired for comparison sake. The 3D engine parameters are: a point in space (x, y, z) and a 3D model. The output, a rendered image of the model, has color and lightening similar to the ones at the moment of acquisition.

2.3. Evolutionary algorithms

Two evolutionary algorithms were implemented. The first is the standard model of Evolutionary Strategy [6] using *uncorrelated mutation with n step sizes* whereas the second algorithm modifies the former. The proposed evolutionary algorithm takes into account the evolutionary strategies of Rechenberg e Schwefel [8] with self-adaptation (self-adaptation is part of evolutionary strategies since 1977 [9]).

Standard Algorithm

Representation

Each member of the population represents a possible solution: a 3D model of the input image. Each phenotype (i.e. 3D model) is formed by a set of vertices, edges and surfaces. All individuals have the same vertices, edges and surfaces, being different in the location of their vertices. So, each genotype (i.e. individual) is represented by a set of vertices in a 3D space given by $(x, y, z) \in \mathbb{R}^3$. Hence, the representation is a sequence of coordinates:

$$(v_1, v_2, \dots, v_n) \equiv (x_1, y_1, z_1, x_2, y_2, z_2, \dots, x_n, y_n, z_n) \quad (3)$$

The *phenotype-genotype* mapping is performed in straightforward manner by arranging the vertices coordinates observing the sequence given, as shown above. As opposed to that, attributing the coordinate values to the model vertices of an individual genotype performs the genotype-phenotype mapping. Furthermore, the proposed evolutionary algorithm uses self-adaptation of parameters, then individual genotype additionally includes the self-adapting parameters (described later).

Evaluation function

The evaluation function aims to assess how well the model represents the object being modeled based on the images provided as output by the 3D Engine. Thus, for every point-of-view received as input, the 3D Engine renders an image of the model. The image comparator yields the difference between the rendered

image and the target one for each point-of-view. Then, all differences are added and the result is divided by the considered number of points-of-view (np), as shown in Equation (4). The result is taken as the fitness value of the individual that should be minimized, since this is an error measure.

$$EF(indiv.) = \sum_{i=1}^{np} \text{diff}(img_{input}[i], img_{rendered}[i]) / np \quad (4)$$

Variation Operators

The variation operators are intermediary and discrete recombination, and mutation by Gaussian perturbation. To decide which type of recombination to be used, we introduce a specific parameter, $\gamma \in [0..1]$, taken as the probability to select intermediary recombination and $(1-\gamma)$ is the probability to employ discrete recombination. The intermediary recombination is suitable for two chromosomes with similar genes values and with far-valued fitness. Discrete recombination is useful when “good” features are in distinct regions of the genotype.

Recombination

The intermediary recombination operator takes two individuals (i.e. parents) to recombine their genes to produce two new individuals (i.e. siblings). The genes of the offspring are obtained as weighted combinations of the genes of their parents. Weights are randomly selected at every new recombination, using a recombination factor $\alpha \in [0,1]$; and are recalculated by Equation (5) to combine genotypes of parents X and Y .

$$\begin{aligned} child_1 &= \alpha \bar{x} + (1 - \alpha) \bar{y} \\ child_2 &= \alpha \bar{y} + (1 - \alpha) \bar{x} \end{aligned} \quad (5)$$

where, \bar{x} and \bar{y} are parents genotypes.

In discrete recombination, the genes of the first sibling are copied from one of the parents, at chance and the genes of the second sibling are complementary values to the first, i.e., the remaining genes in the parents.

Mutation

This operator modifies each gene of the individual by incrementing (or decrementing) them by a given value randomly extracted from the Gaussian distribution $(N(0, \sigma_i))$ with average 0 and standard deviation σ_i : $x'_i = x_i + N(0, \sigma_i)$, (6)

where, x_i is the i^{th} gene of the individual.

The standard deviation σ_i is the mutation step size, a possibly differently value to each gene. This technique is known as *uncorrelated mutation with n step sizes* [6]. In this approach the step sizes σ_i are treated as genes, thus they are inserted into the

chromosome. So the representation is changed as is in Equation (7)

$$(g_1, g_2, g_3, \dots, g_n, \sigma_1, \sigma_2, \sigma_3, \dots, \sigma_n), \quad (7)$$

where g_i s are the genes represented in (3).

As a consequence, the mutation step of each gene, σ_i , is subject to recombination and mutation. The mutation operator is applied first to genes that correspond to the mutation steps, then, the genes that correspond to the coordinates of the model vertices (g_i) are mutated as describe before, by utilizing the new mutation steps.

A log-normal distribution determines variations generating different values to each σ_i and shares the same parameter τ with all step sizes. This value ranges within the interval $\sigma_{\min} \leq \sigma_i \leq \sigma_{\max}$:

$$\sigma_i' = \sigma_i + e^{N_i(0, \tau)} \quad (8)$$

Selection Operators

Parents and survivors of each generation are chosen according to a ranking based selection, as described in [6]. In this mechanism the probability of selection is proportional to fitness and can be adjusted according to a parameter that gives the selection pressure.

Other Parameters of the Evolutionary Algorithm

It is worth mentioning that the algorithm at hand has also other usual parameters namely: population size (μ), number of new individuals created in each generation (λ), initial mutation steps (σ_0), mutation rate (p_m), and recombination rate (p_c).

Modified Algorithm

Representation

The experiments with the standard algorithm suggest that the vertices change occur in a promising direction (in 3D space) instead of wandering around its proximities. So, once the evolution found a good direction for a vertex it should keep evolving according to the direction while the individual fitness improves. Hence, the representation was changed to include an evolution (or mutation) direction for each vertex. A simple representation for a direction vector in 3D space uses two angles: α in relation to the x-axis and β in relation to the y-axis and they are evolved using the former mutation step size. The new individual is represented as follows:

$$\left(\begin{array}{l} x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n, \\ z_1, z_2, \dots, z_n, \alpha_1, \alpha_3, \dots, \alpha_n, \\ \beta_1, \beta_2, \dots, \beta_n, \sigma_1, \sigma_3, \dots, \sigma_n \end{array} \right), \quad (9)$$

where x_i , y_i , and z_i , represent the i^{th} vertex coordinates (the previous g_i 's.). α_i , and β_i are the vector angles and σ_i , represents the step sizes.

As n is now the number of vertices, it values $1/3$ of n in the previous representation (number of coordinates). Additionally the new representation demands only one mutation step size for each vertex instead of the three per vertex. At first glance, this modification increases the dimension of the search space, however, the search is limited to the directions represented by α and β . The coordinates are used only to keep track of the current positions of the vertices. The step sizes can be seen as the magnitudes of the direction vectors.

Mutation

The mutation mechanism is specifically implemented for each part of the chromosome and they are applied in the following order: first the angles, then the step sizes and finally the coordinates are mutated. The mechanism is specified by Equations (10-13):

$$\begin{aligned} \alpha_i' &= \alpha_i + N(0, \sigma_a), \\ \beta_i' &= \beta_i + N(0, \sigma_a), \end{aligned} \quad (10)$$

where σ_a , is a fixed value: the angle mutation step size. The resulting angle is normalized between zero and 2π . The step size mutations follow:

$$\sigma_i' = \sigma_i + e^{N_i(0, \tau)} \quad (11)$$

where τ , is the learning rate and $\sigma_{\min} \leq \sigma \leq \sigma_{\max}$. The

$$x_i' = x_i + \Delta x_i$$

coordinate mutations are: $y_i' = y_i + \Delta y_i$, (12)

$$z_i' = z_i + \Delta z_i$$

where Δx_i , Δy_i , Δz_i are the projections of the direction vector along the x, y and z axes and they can be computed from α_i , β_i , and σ_i as follows:

$$\begin{aligned} \Delta x_i &= \sigma_i \cos(\alpha_i) \cos(\beta_i) \\ \Delta y_i &= \sigma_i \sin(\alpha_i) \\ \Delta z_i &= \sigma_i \cos(\alpha_i) \sin(\beta_i) \end{aligned} \quad (13)$$

Recombination

The recombination has a single change with respect to the angles mean. If one angle being combined is lower than 180° and other is higher, then the resulting angle is the mean between the lower angle and the higher minus 360° , instead of the higher itself. For example, the mean between 45° and 315° must be 0° instead of 180° . This ensures that the mutation direction of the new individual is close to its parents mutation direction instead of opposed as it could happen using traditional mean.

Other Parameters

The others parameters remain the same.

3. Simulations

Simulations included here evaluate the performance of the standard and modified algorithms in reconstructing both very simple and more complex objects.

3.1. Simple object modeling

In this simulation, the ERS was supplied with three different points-of-view of a target object (Figure 2) and a symmetric initial model (Figure 3).

The results (Figure 4) stand for the best solutions after five executions of each algorithm. The analysis of the results would be statistically more significant if more runs were executed, this was not proceeded due to time restrictions.

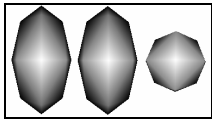


Figure 2
Points-of-view of modeled object

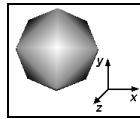


Figure 3
Image of a sphere-shaped object with 18 vertices.

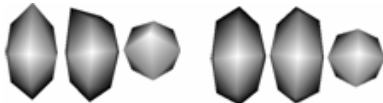


Figure 4 – Standard algorithm (left) and modified algorithm (right) reconstructed models

The strategy parameters shared by the algorithms have the same values shown in Table 1. The parameter values were chosen by the trial and error.

Table 1 – Parameters used in the evolutionary algorithms.

| Parameter | μ | λ | p_m | σ_{\min} | σ_{\max} | σ_0 | σ_a | τ^1 | p_c | Γ |
|-----------|-------|-----------|-------|-----------------|-----------------|-----------------|------------|----------|-------|----------|
| Standard | 20 | 20 | 0.8 | 0.001 | 0.01 | σ_{\min} | - | 0.23 | 1 | 0.5 |
| Proposed | 6 | 6 | 0.8 | 0.001 | 0.01 | σ_{\min} | $\pi/2$ | 0.4 | 1 | 0.5 |

Figure 5 displays learning curves of the standard and modified algorithm. In this picture one can note that the proposed algorithm achieved better results faster, so it is more effective and efficient.

The graphic shown in Figure 6 displays the evolution of the average of the self-adapted mutation step in each generation on both algorithms. It tends to be lower in the final stages due to the proximity of the genes to their best. This situation may cause parents fitter than their children. The reduction in the amount

of step sizes makes easier the self-adaptation along the evolution process.

To compare the meshes obtained using the standard Evolutionary Strategy and the modified one we propose, we measured the distance from each mesh to the target mesh using the Metro tool [10]. The Hausdorff distances obtained are 0.275908 (Standard ES) and 0.093381 (Modified ES).

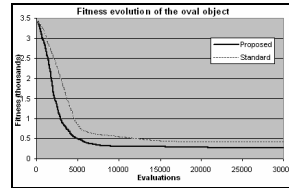


Figure 5
Evolution of the ellipsoid object

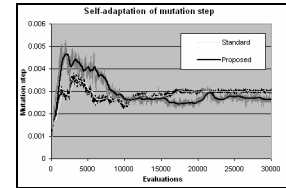


Figure 6
Self-adaptation of mutation step

3.2. More complex objects modeling

A pear- and a face-shaped object are reconstructed from an ellipsoidal object with 66 vertices and from a paraboloidal surface with 121 vertices respectively using the modified algorithm. The initial models, points-of-view supplied and results are shown in Figures 7 and 8.

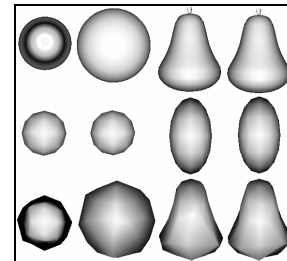


Figure 7 – Pear reconstruction, 4 of 6 given points-of-view, initial model views, reconstructed model views.

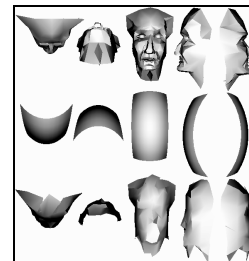


Figure 8 – Face reconstruction, all given points-of-view, initial model views, reconstructed model views

4. Conclusions

The evolutionary approach put forward in this article succeeded in producing models that are able to reconstruct simple 3D objects based on images of themselves.

The number of vertices to reconstruct objects varies with the complexity of the objects to be reconstructed. This is a common feature of the reconstruction methods, the ERS loses performance, in terms of time and accuracy, when facing growth of complexity due to an increase of the search space. The current version of the ERS does not capture fine details of complex objects. This is, certainly, a future research topic for evolutionary algorithms used to reconstruct objects.

A possible solution for solving the problem posed above is to start the search from simple objects and to introduce new vertices as the evolution process identifies that the existing vertices do not model precisely the target object. This can be achieved by using flexible genotypes as they codify the vertices of the searched model.

We also observed that both are important: the number of vertices of the initial models and also the way they are connected to form an object. The choice of an initial model, from which the target object is modeled, launches the start region of the search space for the evolutionary process and delimits the shape of objects that can be tackled. We found that this choice is crucial for the success of the reconstruction because initial populations with phenotypes that are very different from the target object, can hardly lead to solutions near to global optima.

The choice of an initial model is viewed as both: a limitation and an advantage. Firstly, this choice can generate solutions with low amount of information (i.e. images) in which the quality of the solutions can be improved as more information is available. Secondly, such a choice avoids the problem of merging multiple polygonal meshes into a singular polygonal mesh faced by other approaches [1]. This merging process, usually, creates undesired holes in the model in the region of the boundary between two separate views. These undesired holes do not appear in our model once it uses deformable surfaces, whose vertices are moved without changing the connections determined in the initial model.

The number and quality of the *points-of-view* (given images) to be supplied to the system must allow modeling the target object. If few images are supplied, the evolution process may find solutions that satisfy such given views however are not accurate models of the target object. However, the higher the number of *points-of-view* supplied, the more computational resources are required for comparing images and evaluating fitness.

Therefore, a fine balance between number of points-of-view and the details they encompass is essential for producing an accurate output within acceptable time.

Finally, the proposed modifications in the standard algorithm can be applied to any problem, which the genes can be grouped in subspaces. This makes it easier to the evolutionary algorithm to find better values for the self-adapted mutation step sizes.

5. References

- [1] R. J. Campbell, P. J. Flynn: "A survey of free-form object representation and recognition techniques" *computer vision and image understanding* 81 (2): 166-210 Feb 2001
- [2] M. Weidenbach, S. Trochim, S. Kreutter, C. Richter, T. Berlage, G. Grunst, "Intelligent training system integrated in an echocardiography simulator," *Computers In Biology And Medicine* 34 (5): 407-425 Jul 2004.
- [4] M. Vasilescu and D. Terzopoulos, "Adaptive Meshes and Shells," *Proc. IEEE Computer Vision and Pattern Recognition Conf.* '92, pp. 829-832, 1992.
- [5] T. McInerney and D. Terzopoulos, "A Finite Element Model for 3D Shape Reconstruction and Nonrigid Motion Tracking," *Proc. Fourth Int'l Conf. Computer Vision*, pp. 518-523, 1993.
- [6] Y. Chen and G. Medioni, "Surface Description of Complex Objects from Multiple Range Images," *Proc. IEEE Computer Vision and Pattern Recognition Conf.*, pp. 153-158, 1994.
- [7] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing Springer*, ISBN, 2003.
- [8] VRML Mall - a 3D gallery, internet link: <http://www.ocnus.com/models/>
- [10] H. G. Beyer, H. P. Schwefel. "Evolution Strategies: A comprehensive introduction". *Natural Computing*. 1:1 pp.3-52, 2002.
- [11] H. P. Schwefel. "Numerische Optimierung von Computer-Modellen mittels der Evolutionstrategie", Vol 26 of ISR. Birkhaeuser, Basel/Stuttgart, 1977
- [12] P. Cignoni, C. Rocchini, and R. Scopigno. "Measuring error on simplified surfaces." In David Duke, Sabine Coquillart, and Toby Howard, editors, *Computer Graphics Forum*, volume 17(2), pages 167-174. Eurographics Association, 1998.