Evolvable Modeling: structural adaptation through hierarchical evolution for 3-D model-based vision*

Thang C. Nguyen, David E. Goldberg†, Thomas S. Huang
Beckman Institute and Coordinated Science Laboratory
†Department of General Engineering
University of Illinois, Urbana, IL 61801, USA
cthang@uirvld.csl.uiuc.edu, goldberg@vmd.cso.uiuc.edu, huang@uicsl.csl.uiuc.edu

Abstract

We present a system that lets 3-D models evolve over time, eventually producing novel models that are more desirable than initial models. The algorithm starts with some crude models given by the user, or randomly-generated models from a given model-grammar with generic design rules and loose constraints. The underlying philosophy here is to gradually *evolve* the initial models into structurally novel and/or parametrically refined models over many generations. There is a close analog in the evolution of species where better-fit species gradually emerge and form specialized niches, a highly efficient process of complex structural and functional optimization. Simulation results for model jet plane design illustrate that our approach to model design and refinement is both feasible and effective.

Introduction

The motivations for our work are:

a. Currently, some existing systems (e.g., [Goa83], [Ike88] and others with various degrees of sophistication, [Bro81], [Ume93], [Zha92]) allow us to generate recognition strategies/programs, once we have an appropriate 3-D model of the target. A bottleneck in the development of these systems, however, is the high costs of good model base design, indexing and coding.
b. Real-world objects are very diverse. For cars, planes, chairs, etc. there is much greater variability/parametrization than could be captured by hand-crafting of models. Vision system developers often do not have at hand the design blueprints for their objects of interest. And even man-made objects such as airplanes do have their evolution trend of progress towards higher complexity and specialization over the course of time. Thus if a system can automatically evolve its model base then it can achieve a higher degree of autonomy in dealing with new-object cases, with lower requirements in human "teaching" or tedious hand-crafting.

Though still limited, GA/GP, or genetic-algorithms -programming, has been applied successfully to a few vision and graphic applications, such as: segmentation of intensity images by Bhanu et al [Bha90], segmentation of range images by Meygret et al [Mey92], pattern recognition [Cal87], and automatic generation of abstract color art [Sim91]. Notably, Hill et al [Hil92] applied GA to generate flexible 2-D templates for recognition of left ventricles in echocardiograms. However, we have not been successful in looking for published works on 3-D model-based recognition with model evolution.

Evolvable 3-D modeling with multi-level GA/GP

Our approach is to define 3-D geometric models in such a way that one can crossover (or "mate"), in the spirit of GA/GP, between 2 "parent" models and get 2 new models (offsprings), sometimes with new characteristics not possessed by either parents. [Gol89] [Koz92] The underlying principles of GA/GP on which we built this platform can be summarized as follows:

a. Grammar-based, structured object modeling: a plane is defined structurally as composed of a body, wings, engines, stabilizer tails and vertical rudder. Each of these structure is composed of geometric primitives like cylinders, cones, ellipsoids and polygons. Our implementation allows us to start with a symbol "Plane" and let the grammar expands fully, and then all terminals evaluate to function calls that eventually give a set of all 3-D geometric primitives ready for graphical display or printout. Important configuration variations like engines and tails placement (tip-of-rudder or tail-body) are also incorporated into the grammar. We are thus able to "crossover" structural elements among the symbolic designs.
b. Parametrization of numerical model feature values, feature relation coefficients, and inter-feature

* This work has been supported by the National Science Foundation NSF grant IRI-89-02728.

constraints. For example, we define many numerical parameters in terms of other parameters. The numerical parameters are coded into binary strings that can be crossed and give children binary strings.

c. Performance feedback, either from human user or automatic recognition process, contributes to shape the path of model evolution: current models that are more similiar to desirable models have more chance of surviving and evolving further. Evolution processes have been shown to be much more efficient than random search, and also a robust approach to optimization [Goldberg89].

With respect to genetic operations (crossover and mutation), each of our models has 2 levels:

- ☐ Structural level: with a production grammar and symbolic parameters. A model design expressed at this level has the form of a tree with symbolic parameters. A crossover at this level is performed by exchanging 2 subtrees of the same functionality: e.g., 2 *Wings* subtrees.
- ☐ binary string level: for the coding of numerical parameter values. At this level, a crossover between 2 models can be performed by crossing 2 binary strings, both of which refer to some common parameter, e.g. *WingHalfSpan*.

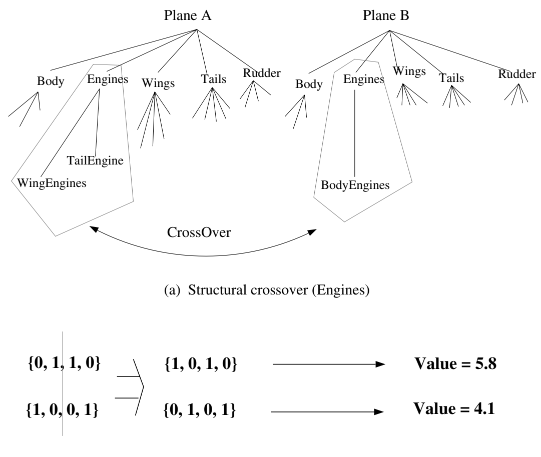


Figure 1. Crossover between 2 planes at the structural level (Engines), and binary string level for numerical parameter values.

In practice, some substructure is picked randomly, from among all substructures of a symbolic design, for crossover or mutation. This has the same effect of hierarchical sampling of both structures, like GP, and parameters (with GA string operations).

Experiments on modeling of jetliners through simulated evolution

We present results of a typical run of interactive design of jetliners. However, with appropriate augmentation, the same system here can be embedded as a hypothesis generator and model acquisition module within a fully automatic object recognition system.

The *planeStructure*, such as those in figure 1(a), is constructed according to a *planeGrammar*, whose rewriting rules are as follows:

- ☐ Plane—>DesignPlane[Body, AllEngines, Wings, Tails, Rudder]
- ☐ Body—>SketchBody[midbodyLength, tailLength, bodyDiameter]
- ☐ AllEngines—>EnginesPick[NumOfEngines[Random[ ]]]
- ☐ WEngines—>WEnginesAt[WEngineFromBase]
- ☐ BEngines—>BEnginesAt[BEnginesFromEnd]
- ☐ Wings—>SketchWings [wingHSpan, wingbaseWidth, wingXbaseWidth, wingbaseleadPointX, wingElbowK, wingSweep, wingtipWidth]
- ☐ Tails—>SketchTails[tailHSpan, tailbaseWidth, tailtipWidth, tailSweep, tailsMount]
- ☐ Rudder—>SketchRudder[rudderbaseWidth, ruddertipWidth, ruddertipHeight, rudderSweep]

We allow each of our numerical parameters to take on its value within some constraint interval, into which its binary string will be mapped and interpreted. There are 19 parameters which participate in GA-style genetic operations. Most of these parameters are not dependent on some other parameters, except *wingbaseleadPointX, wingElbowK, wingEngineFromBase,* and *bodyEngineFromTailEnd*. Following is a partial list of the parameter constraints:

- {midbodyLength, IsWithin, {6.0, 9.0}}
- {bodyDiameter, IsWithin, {0.8, 1.2}}
- {wingHSpan, IsWithin, {4.5, 6.8}}
- {wingbaseWidth, IsWithin, {1.8, 3}}
- {wingbaseleadPointX, IsWithin, {0.52, 0.65}, midbodyLength}
- {wingElbowK, IsWithin, {0.15, 0.35}, wingHSpan}
- {wingEngineFromBase, IsWithin, {0.25, 0.5}, wingHSpan}
- {bodyEngineFromTailEnd, IsWithin, {-0.15, 0.05}, midbodyLength}

A constraint interval by itself does not give a parameter value directly, but given a N-bit binary string *S* for any independent parameter and its corresponding parameter constraint interval {lowerpoint, upperpoint}, the parameter value is determined as follows:

$$val = lowerpoint + \frac{(upperpoint - lowerpoint) \sum_{k=0}^{N-1} 2^k * S_k}{2^N - 1}$$

where S_k is the kth bit's value (0 or 1). This is just a simple linear mapping of the binary string S into the interval, including the lower end point.

In the case of a parameter that depends on some other parameter(s), the parameter's value is generally computed as an expression. For example, suppose we have a string 0110 for *wingEngineFromBase*, and suppose the value of *wingHSpan* is 4.96, one first computes the string's value to be 0.35, and then one uses that 0.35 as a coefficient to compute the value for *wingEngineFromBase* to be 0.35 * 4.96, which is 1.736. The same is true of the other dependent parameters.

There are also parameters whose values are kept constant, and those relations between parameters, which are actually equations with evolvable variables (those parameters coded into GA binary strings). We called these parameters, in rule form, *HiddenParameters* (two of them, *BodyTailsMount* and *RudderTailsMoun*t, are 3-D vector-valued functions). Even though *HiddenParameters* do not directly participate in any genetic operations, those that depend on the "evolvable" 19 parameters do have their values indirectly "evolvable". Here are some of them:

□rudderbaseHeight—> tailLength * sin[0.1] + 0.02
□rudderleadPointX—> rudderbaseWidth – tailLength + 0.05
□BodyTailsMount—> { tailbaseWidth – tailLength + 0.05, 0, rudderbaseHeight }
□RudderTailsMount—> { rudderleadPointX – rudderSweep + 0.05, 0, ruddertipHeight – 0.1 }

For simplicity of the presentation and discussion of our results here, each generation consists of only 4 jetliner models, probably the barest minimal size for any kind of simulated evolution. The overall flow of the evolutionary modeling process is as follows:

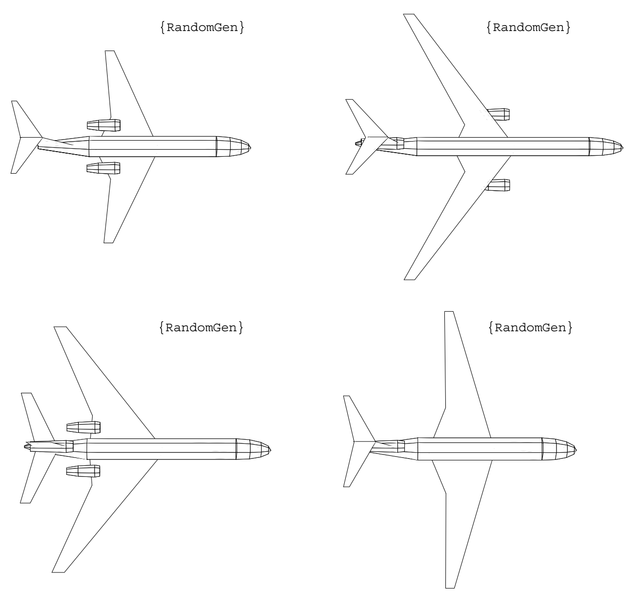
a. The first population of models consists entirely of randomly generated models from the model grammar, model primitives, parametric relations and constraints.
b. The model designs are then interpreted graphically and rendered. Each jetliner model is then given a score, judged by the user (or by some internal performance measure, if within an object recognition system).
c. The two models with highest scores are then selected for crossover, or "mating". One randomly selected model (among the current 4) is then "mutated". And one new model will be randomly generated (in the manner of the first-generation models.) This process creates the new generation of 4 models.
d. The above loop in (b) and (c) is continued until at least one model is given a score higher than a threshold of acceptability (say, any model to be accepted has score > 100.) Alternatively, a maximum number of generations can be set beforehand, and many runs can be tried.

We keep a simple format for quick comprehension of the process. See figure 2: there are 4 populations, from the 0th generation to the 3rd generation of the evolution process. In each population (except the 0th generation), models at positions 1 and 2 are children of a crossover operation (sexual production) from the last generation; model at position 3 is a mutated offspring; and model at position 4 is a new randomly generated model. Incidentallly, we view a mutation as a crossover with some random unknown model, which is appropriate in our framework. Figure 3 shows 4 views of the best model found.

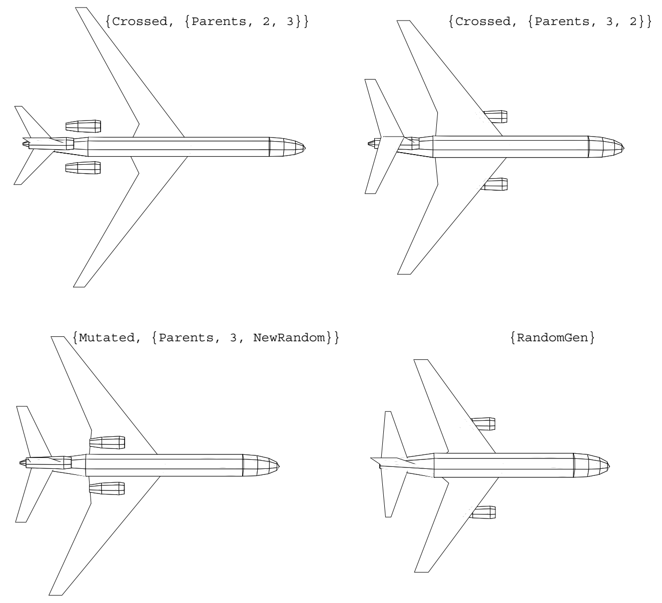
Comments and Conclusions

We have demonstrated that it is feasible to evolve 3-D jet models to search for better models, starting from random models. There are many ways to apply our evolvable modeling system. This is one method to let a system learns to acquire new 3-D models with only implicit and partial, non-specific information (human user or recognition modules need only give some "fuzzy" score to the quality of any model proposed.) In many realistic situations, the evaluation criteria of a model cannot be put down very concretely, especially with an object unknown to even the system designer. We believe our platform to be among the first to perform 3-D model evolution that have a high degree of realism for model-based recognition, despite having to simplify the graphical rendering parameters to avoid clutters at a small display scale.

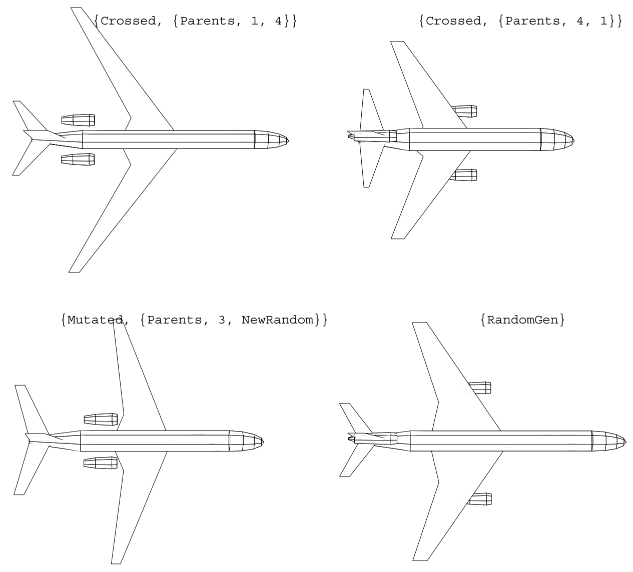
Many simulations of this nature have been run and the results found to be very interesting, with some novel and pleasing designs. On the other hand, conventional jetliner models are just as conveniently generated. The best feature of this platform is its high autonomy, thanks to the underlying philosophy of evolutionary processes.



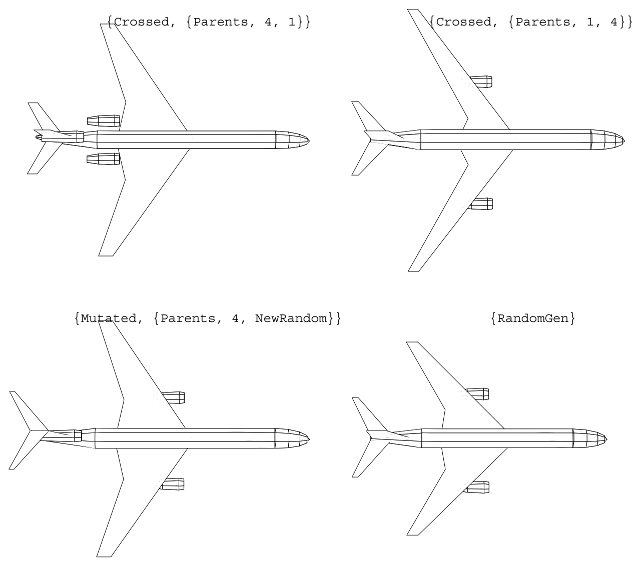
0th generation: all random jets


{Crossed, {Parents, 2, 3}}
{Crossed, {Parents, 3, 2}}
{Mutated, {Parents, 3, NewRandom}}
{RandomGen}

1st gen: planes 1,2 from crossover of 2, 3 in generation 0.


{Crossed, {Parents, 1, 4}}
{Crossed, {Parents, 4, 1}}
{Mutated, {Parents, 3, NewRandom}}
{RandomGen}

2nd gen: planes 1, 2 from planes 1, 4 of 1st gen.


{Crossed, {Parents, 4, 1}}
{Crossed, {Parents, 1, 4}}
{Mutated, {Parents, 4, NewRandom}}
{RandomGen}

3rd gen: planes 1, 2 from planes 1 and 4 of 2nd gen.

Figure 2. A typical run to evolve jet planes over 4 generations. In each generation, planes are numbered left to right, top to bottom, from 1 to 4.

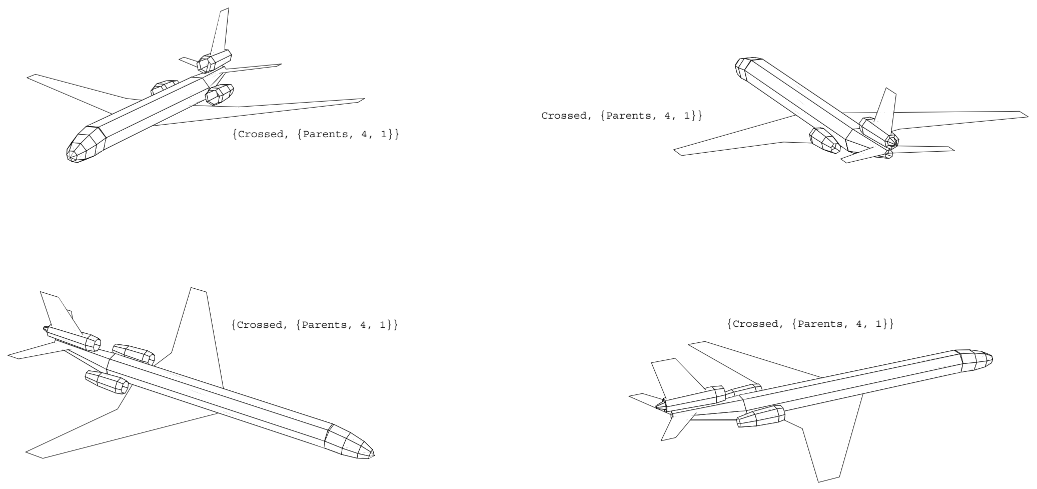

{Crossed, {Parents, 4, 1}}
Crossed, {Parents, 4, 1}}
{Crossed, {Parents, 4, 1}}
{Crossed, {Parents, 4, 1}}

Figure 3. The best plane so far, plane #1 in the 3rd generation above. It has the following chracteristics: moderate body length, relatively small body diameter, moderate wing span, three engines, all mounted at the tail end of the body, a body-mounted stabilizer tails (as opposed to rudder-tip, high mounting position like plane #3 of generation 3).

References

[Bha90] Bhanu, B., Lee, S., and Ming, J., Self-optimizing control system for adaptive image segmentation, *Proc. Image Understanding Workshop*, 1990, pp. 583–596.

[Bro81] Brooks, R. A., Symbolic Reasoning Among 3-D Models and 2-D Images, *Artificial Intelligence*, vol. 17, 1981, pp. 285–348.

[Cal87] Calloway, D., Using a genetic algorithm to design binary phase-only filters for pattern recognition, *2nd Intl. Conf. on Genetic Algorithms*, 1987, pp. 422–429.

[Esp92] Esposito, F., Malerba, D., and Semeraro, G., Classification in Noisy Environments Using a Distance Measure Between Structural Symbolic Descriptions, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 3, March 1992, pp. 390–402.

[Goa83] Goad, C., Special Purpose Automatic Programming for 3D Model-based Vision, *Proc. Image Understanding Workshop*, 1983, Virginia, USA, pp. 94–104.

[Gol89] Goldberg, D. E., Genetic Algorithms in search, optimization and machine learning, *Addison-Wesley*, 1989.

[Hil92] Hill, A., and Taylor, C. J., Model-based image interpretation using genetic algorithms, *Image and Vision Computing*, vol. 10, no. 5, June 1992, pp. 295–300.

[Ike88] Ikeuchi, K., and Kanade, T., Automatic Generation of Object Recognition Programs, *Proc. of the IEEE*, vol. 76, no. 8, August 1988, pp. 1016–1035.

[Koz92] Koza, J., Genetic Programming, on the programming of computer by natural selection, *MIT Press*, 1992.

[Mey92] Meygret, A., Levine, M. D., and Roth, G., Robust Primitive Extraction in a Range Image, *11th Intl. Conf. on Pattern Recognition*, 1992, vol. III, pp. 193–196.

[Sim91] Sims, K., Artificial Evolution for Computer Graphics, *Computer Graphics*, vol. 25, no. 4, July 1991, pp. 319–328.

[Ume93] Umeyama, S., Parametrized Point Pattern Matching and Its Application to Recognition of Object Families, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 2, February 1993, pp. 136–144.

[Zha92] Zhang, S., Sullivan, G. D., and Baker, K. D., Using Automatically Constructed View-Independent Relational Model in 3D Object Recognition, *Proc. 2nd European Conference on Computer Vision*, Italy, pp. 778–786.