A generative design system based on evolutionary and mathematical functions†

Liu Xiyu
Design Technology Research Centre, School of Design
The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong SAR
Email: sdxyliu@polyu.edu.hk
and
School of Information and Management
Shandong Normal University, Jinan, Shandong 250014, P.R. China
Email: xyliu@sdnu.edu.cn

John Hamilton Frazer
Swire Chair Professor,
Director of the Design Technology Research Centre
School of Design
The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong SAR
Email: John.Frazer@polyu.edu.hk

Tang Ming Xi
Design Technology Research Centre, School of Design
The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong SAR
Email: sdtang@polyu.edu.hk

Abstract

Previous work by Professor John Frazer on Evolutionary Architecture provides a basis for the development of a system evolving architectural envelopes in a generic and abstract manner. Recent research by the authors has focused on the implementation of a virtual environment for the automatic generation and exploration of complex forms and architectural envelopes based on solid modelling techniques and the integration of evolutionary algorithms, enhanced computational and mathematical models. Abstract data types are introduced for genotypes in a genetic algorithm order to develop complex models using generative and evolutionary computing techniques. Multi-objective optimisation techniques are employed for defining the fitness function in the evaluation process.

1. Introduction

A related paper [11] at this conference reviewed and described the theoretical foundations of the research led by Professor John Frazer in the past in the UK and in particular more recently in Hong Kong, on generative and evolutionary architecture design. In this paper, we focus on the technological implications and implementation techniques of a system for the design and visualisation of both abstractive 3D forms and domain specific architectural envelopes based

† Research sponsored by the Research Fellow Matching Fund Scheme 2001 (No. G.YY.34) of the Hong Kong Polytechnic University, and the Governmental Young Scientist Award Foundation of Shandong, P.R. China.

on an integration of three main computational techniques. In this paper we discuss these techniques and present (1) computational tools for creating and exploring alternative complex forms, (2) stimulating the process of generating abstract but novel design concepts using generative design techniques, (3) linear and non-linear algorithms for modifying abstract forms to obtain complex forms through spatial or conceptual transformations. The facilities for providing these functions are integrated in a new system based on enhanced 3D solid modelling techniques with complex mathematical functions. The system kernel is compatible with object-oriented technology and 3D solid modelling and surface modelling standards. The system is demonstrated in an evolutionary architecture paradigm with a focus on how to generate visionary and creative forms. The complex forms generated and visualised using the developed system are promising. Our system has been implemented based on an integration of ACIS™ 3D solid modelling kernel and MatLab™ with a C++ graphical user interface. The integration of generative and evolutionary computation techniques with 3D solid modelling techniques provided a solid foundation for us to develop more domain specific applications. The system is fully compatible with commercial CAAD tools and systems, as well as rapid prototype facilities. Theoretical concepts of sophisticated surfaces and envelopes based on a library of basic building blocks of complex form have been built using evolutionary techniques and partial ordering theory of non-linear analysis.

A new type of genetic algorithm is also studied for our generative design system. We extended the classical powerful techniques from modern non-linear analysis theory to selection and optimisation of GA. These techniques included topological spaces and partial ordering. A Zorn Lemma type of iterative procedure is introduced. This attempt partially overcame the difficulty in implementing effective automatic selection in the application of genetic algorithms.

2. Mathematical models and 3D shapes

Nowadays, one of the most significant ways to understand a mathematical model is through computer visualization. However, due to the fully non-linear nature of many functions, it is not an easy task to develop accurate shapes for general non-linear functions. One way to solve this problem is the use of finite element analysis methods. There are several approximate techniques for non-linear functions, the simplest of which is a planar piece. More accurate techniques included NURBs approximation, polynomials approximation and others. It should be noted that it is always difficult to find a good balance between a better approximation and the acceptable computing time.

With the help of a solid modelling kernel with libraries of many sound geometric transformation and reasoning methods, we developed 3D solid model visualisations for complex functions in this project. A prototype system has been implemented based on an integration of ACIS 3D solid modelling kernel and MatLab with a C++ graphical user interface. Our basic geometrical objects for approximation are NURBs surfaced units. Our system is fully compatible with any commercial CAAD tools and systems, as well as rapid prototype facilities. A large number of object-oriented components of sophisticated surfaces and envelopes have been built. In particular, complex forms are classified as linear, quadratic, trigonometric function, exponential functions, root functions compounded functions, rotations, sphere and cylinder co-ordinates, implicit function. Computational mechanisms have also been developed with which these basic data structures and components can be visualised, combined or split to allow new data structures or new forms to be derived using generative techniques.

3. Simulating architectural envelopes

In this paper, our main concern is to generate architectural envelopes with their outlines determined by mathematical functions and evolutions. We use a multi-coding schema to represent the phenotypes. That is, we use continuous schemas with continuous functions, and discrete schemas with discrete functions.

It is well known that one of the most important and difficult problems in evolutionary design applications is the appropriate definition of the fitness function. It is this function that determines the selection and optimisation of the evolution and the final solution. In the literature, many authors used artificial selection techniques, which indeed helped to solve part of this problem. However, in a design application, exploration with artificial selection and optimisation with natural selection need to be combined in order to support the process of design from under-constrained design space of abstract concepts to a highly constrained space with well-defined variables and evaluation criteria. A wide range of design problems can be supported using generative and evolutionary techniques

To solve this problem, we adopted a multi-objective fitness function for optimisation, and a partial ordering technique derived from non-linear analysis to represent the complicated relationships among the candidate solutions from in populations generated during the process of evolution.

Figure 3.1 is a diagram of the functional components of the system we have implemented.

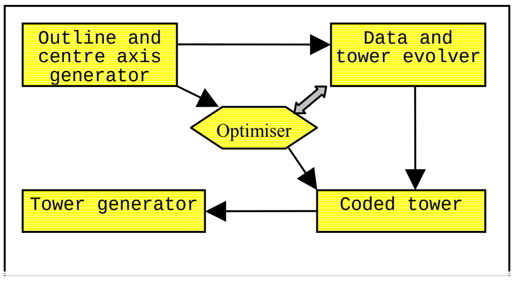

Fig 3.1 System functional components

4. Object Library and object class definition

We used an object-oriented representation in the system implementation. There are 47 object classes in the system object library including user interface classes. Among them, 17 classes are modelling classes. The base modelling class is CModel. Two derived classes are CAModel and CTower. There are three classes derived from CTower and 10 classes from CAModel. One of the CAModel derivations is CGABase, from which some evolution classes such as CGAModel and CANNModel are derived.

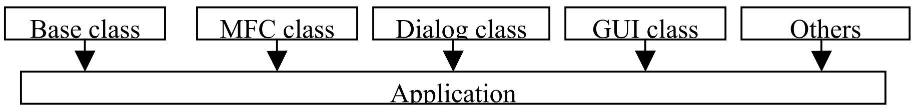

Fig 4.1 Class hierarchy

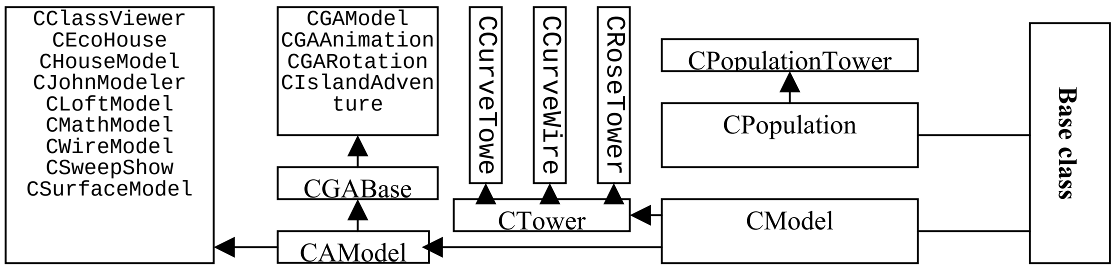

Fig 4.2 Class structures

The base class CModel provides interaction with the Acis 3D kernel and the basic file access. This class has two derivations, CAModel and CTower. The CTower class is the rendering encapsulation of the Acis APIs for the construction of 3D solid model classes. Another derivation is CAModel. In this class, useful encapsulations of rendering functions, for example, texture, colour, cutting etc, are integrated. Common operations such as SAT file save and refresh are also implemented in this class. Some geometrical construction functions are also encapsulated in this layer.

The main working classes in this project are at the next layer. The two derivation lines are CTower and CAModel. Along the first derivation line, there are three classes. The first class is CCurveTower. Below are the main operations for this class:

Table 4.1 Operations of class CCurveTower

```
double outline(double, double, double);
double outlineAxis(double, double, double);
void Create(double, double, double, int, int);
void Create(double, double, double, int, int, double*, double*);
void BuildCone(double, double, double, double, double, double);
void BuildConeTwo(double, double, double, double, double, double);
void BuildHat(double, double, double, double);
void BuildBase(double, double, double, double);
```

The construction of tower models is implemented in this class. The model is controlled by two functions. One is for the outline of the tower and the other is the centre curve. Figure 4.2 illustrates two example towers constructed using this class definition.

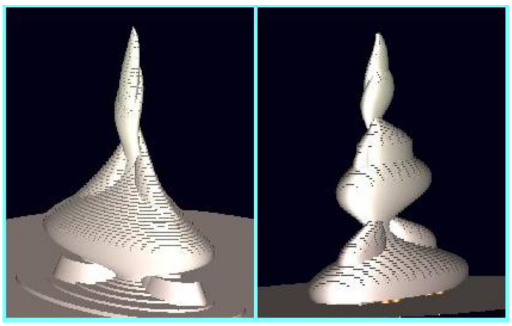

Fig 4.2 Towers created by class CcurveTower

The second class is CCurveWire. This is a class for constructing the wire-frame around the tower. Apart from the operations similar to the previous class, a new function called the bone-construction is provided as follows.

Table 4.2 Operations of class CCurveWire

void BuildBone(double*, double*, int, int);
void BuildBone(double, double, double, double, double, double, int, int);
void BuildBaseWire(double, double, double double);

The examples of models created using this class.

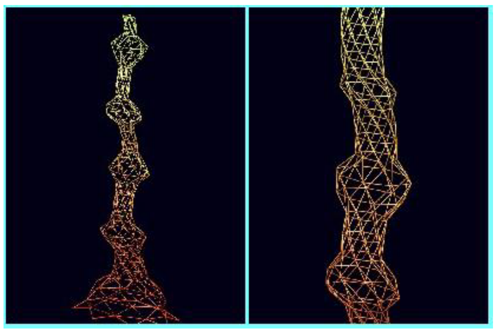

Fig 4.3 Wire frame created by the class CCurveWire

Another class is CRoseTower. The main functionality of this class is to create towers with more than two columns. A typical example is as follows.

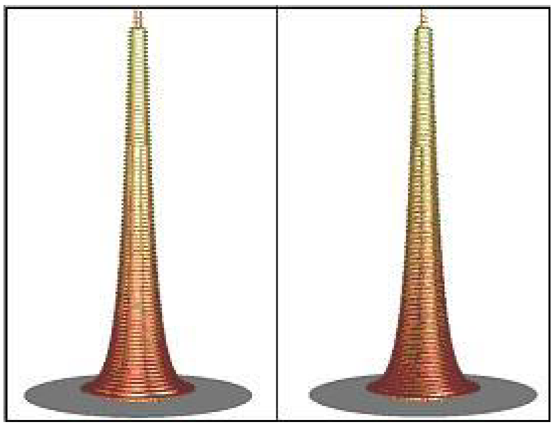

Fig 4.4 Five column tower and three column tower

A subclass at the second layer is CAModel. There are more class derivations than the class CTower. In fact, this class is the most important one in our system. And there are 10 derived classes from CAModel. Most of the classes perform a specific kind of architectural or mathematical model construction task.

Among these ten classes, CJModeler is one that focuses on mathematical modelling. Our main approach to complex form modelling is to combine discrete functions with NURBs surfaces. Some of the operations in this class are shown in the following table.

Table 4.3 Some operations of class CJModeler

```
void BuildP129SliceWire(double, double, double,   double, double, int, EDGE*&);
void BuildSweepPiece(double, double, BODY*&, BODY*&);
void BuildP132WingPiece(double, double, double, double, double, double, double, int, double*);
void BuildP127Piece(double, double, double, double, double, double, double* clColor);
void BuildP121TopPiece(double, double, double, double, double, double, double, double*);
void BuildLoftPiece(int, EDGE**, BODY*& my_body);
double nonSurface(double, double, double, double, int);
void BuildP123Wire(double, double, double, double, double,  double*, int nFlag);
void BuildPipeFromWire(position&, position&, EDGE*&, double, double* clColor, double xRotate = 0);
void BuildKuenPipeFromWire(position& start, position& end, EDGE*& my_edge,
            double radius, double* clColor, ENTITY*& my_body);
void BuildKuen86Wire(double radius, double v, double* clColor, int nFlag=1, int nWireFlag=0, int nPlusFlag=0);

//projective functions
double FProjective1(double x, double y, double z, int);
double FProjective2(double x, double y, double z, int);
double FProjective3(double x, double y, double z, int);
void   BuildProjectiveWire(int nFlag, double radius);
void BuildCoil(..);
void BuildBone(..);
void BuildBaseWire(..);
```

5. Genetic algorithms and evolutionary models

Having become widely used for a broad range of optimisation problems in the last ten years, Genetic Algorithm has been described as a "search algorithm with some of the innovative flair of human search". Genetic Algorithms are today renowned for their ability to tackle a huge variety of optimisation problems (including discontinuous functions), for their consistent ability to provide excellent results and for their robustness. Natural evolution acts through large populations, which reproduce to generate new offspring that inherit some features of their parents (because of random crossover in the inherited chromosomes) and have some entirely new features (because of random mutation). Natural selection (the weakest creatures die, or at least do not reproduce as successfully as the stronger creatures) ensures that, on average, more successful populations are produced in each new generation than less successful ones.

Any evolutionary architectural models require architectural concepts to be described in the form of genetic codes. Then these codes are mutated and developed by computer programs into a series of models called populations. While models are evaluated by optimisation or selection sub-systems, the codes of successful models are constantly picked up until a particular stage of development process is reached.

In order to manipulate a complex model with a generative and evolutionary program it is necessary to define the followings: a genetic code script, rules for the development of the code, mapping of the code to a virtual model and, most importantly, the criteria for selection.

The representation of phenotypes is a fundamental element of any evolutionary system. In design applications, phenotypes represent designs, which formulate possible solutions to be evolved by the system. Moreover, phenotype representation plays a significant role in determining the size and complexity of the genotype. The two main 3D representation methods are surface representation (or boundary representation) and constructive solid geometry (CSG). The first method typically uses combinations of equations and control points to specify shapes, while CSG combines different primitive shapes to form more complex shapes. There is a third of the commonly used solid representation called spatial partitioning. This is to decompose a solid into a collection of smaller, adjoining, non-intersecting solids that are at lower primitive level than the original solid. There are a number of variations including: cell decomposition, spatial-occupancy enumeration, octrees, and binary space-partitioning trees.

Primarily, our model in this paper is the boundary representation model. In this representation, architectural envelopes are represented by mathematical data of the main model geometry. Changing the surfaces of a model is achieved through adding or subtracting mathematical data. In accordance with the mathematical functions type data, we use a cell division model. A cell division model is based on the structure of a living object. As in nature, the shape of a living object is constructed from the basic genetic information in the cells and organisms. The genotype contains information that is the basic construction unit of everything, called the chromosome. Chromosomes form proteins and other large molecules. Chains of molecules will form tissues and organisms to form the whole body. In a natural environment, development begins with the chromosome, which forms the base. Then a number of smaller cells are constructed. Large cells are resulted from joining and other operations and form a multi-cellular structure. In a word, the cell division models simple divide the whole process of involution into basic units and operations.

For a better description of the combination of functions, we use a jelly model as an example. This is a derivation of the cell model. A layer called the jelly layer is added to the model to represent a compound structure. We use this model to represent functions in various combinations. Basically, the model has three layers, that is, the gene layer, the cell layer and the jelly layer as shown in the next diagram.

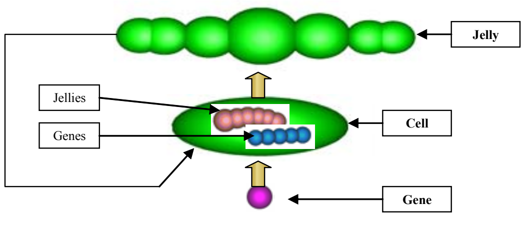

Fig 5.1 A jelly model

It is clear that the above model is like the molecular structure in biology. Jelly plays the role of a large molecule, and is composed of cells. In a cell, there are genes and other small molecules (jelly). The smallest unit in this chain is gene that forms the base of the three structures.

The second model in our system is the discrete model. The basic structures in this model are section and outline data structures. Each of these two outlines is an ordered set of double numbers, with auxiliary data indicating steps and size. To eliminate the discontinuities caused by the data structure, a mollifying operation is introduced.

Mollifying operations act as function smoothers. They can smooth a discontinuous function by substituting the value at one point by some average in a neighbourhood. In the one dimensional case, a mollifying operation is a non-negative, real-valued function $J \in C_0^\infty(R)$ such that

(i) $J(x) = 0$ if $|x| \geq 1$, and

(ii) $\int_R J(x)dx = 1$

An example of a mollifying operation is the following function:

$$J(x) = \begin{cases} ke^{-1/(1-|x|^2)}, & \text{if} \quad |x| < 1 \\ 0, & \text{if} \quad |x| \geq 1 \end{cases}$$

where

$$\frac{1}{k} = \int_{|x|<1} e^{-1/(1-|x|^2)}dx$$

Take $\varepsilon > 0$ and $J_\varepsilon(x) = \varepsilon^{-1} J\left(\frac{x}{\varepsilon}\right)$. Then the convolution

$$J_{\varepsilon^*}(x) = \int J_\varepsilon(x-y)u(y)dy$$

is the mollification of the function *u*(*x*).

Finally, the discrete model can be illustrated as follows.

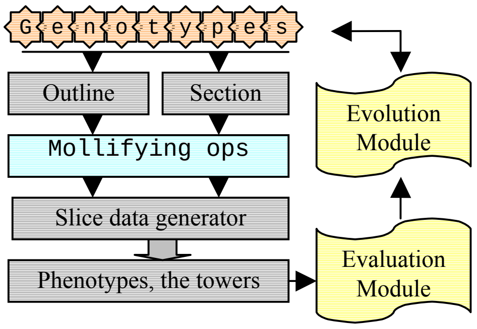

Fig 5.2 Discrete model

6. Multi-objective optimisation

In design systems, there are two main kinds of optimisation, that is, the artificial and automatic optimisation. Within the context of evolutionary design, the first class is usually called explorative evolution, which uses evolution as an explorer, not as an optimiser. In this case the optimisation process is controlled by human designers and the evolutionary system is used to help with the exploration of many possible solutions, so as to provide inspirations and to identify the range of useful solutions.

A modified version of this explorative evolution is an automatic process guided by natural selection defined as fitness function. By telling the computer the desired function in the form of a set of evaluation routines, but not anything about the design itself, the user is removed from the loop and the computer can find the final solutions to design problems. Therefore, a multi-objective optimisation function is needed in this process. Accordingly, the fitness functions must calculate appropriate information about the individual, and then use this information to calculate how well each individual satisfies particular criteria. In short, the fitness functions are a map from each individual to a point in high-dimensional Euclidean space. In this case, partial ordering is an effective method to represent relations of points. In this section we present two definitions in spaces with and without linear structure.

Definition 6.1 Let E be a set and Σ be a subset of $E\times E$. We will call the subset Σ a partial ordering provided that the following properties are satisfied.

(1) $(x,x)\in E\times E$, where $x\in E$.
(2) $(x,y)\in E\times E$, $(y,x)\in E\times E$ imply $x=y$.
(3) $(x,y)\in E\times E$, $(y,z)\in E\times E$ imply $(x,z)\in E\times E$.

Definition 6.2 Let E be a linear space and P is a nonempty convex set. We will call the set P a cone if the following properties are satisfied.

(1) If t is a nonnegative number and $x\in P$, then $tx\in P$.
(2) If $x\in P$, $-x\in P$, then $x=0$.

Cone is an important tool in the study of non-linear problems and positive solutions of differential equations. It is one the three main methodologies in modern non-linear analysis. In [7] the reader can find detailed theory of cone and applications. In general, most of the partial orderings in applications in analysis are derived from cones.

Let P be a cone. Define

$$x \le y \Leftrightarrow x - y \in P$$

Then it is easy to show that the above definition is a partial ordering. In partial ordering spaces, it is often convenient to define a metric. Then we can get a measurable property from a descriptive concept. We call such a measure the Hilbert projective distance. Let P be a cone, and E a partial ordering linear space, $x, y\in E$. If there are positive numbers t and s such that

$$x - ty \in P, \quad y - x/s \in P$$

Then from the definition of partial ordering we have

$$ty \le x \le sy$$

Now we define

$$M\left(\frac{x}{y}\right) = \inf\{\lambda > 0 : x \le \lambda y\} \quad , \quad m\left(\frac{x}{y}\right) = \sup\{\mu > 0 : \mu y \le x\}$$

Definition 6.3 We call the following function the Hilbert projective distance of the two points x, y:

$$\rho(x,y) = \ln M\left(\frac{x}{y}\right) - \ln m\left(\frac{x}{y}\right)$$

Similarly we call the next function be the Thompson distance of x, y:

$$d(x,y) = \ln\left(\max\left\{ M\left(\frac{x}{y}\right), \quad M\left(\frac{y}{x}\right) \right\} \right)$$

Examples Now we consider a simple example. Let E be the linear space of n-dimensional vectors, and let P be the cone of points with non-negative elements, i.e., x=($x_1,x_2,...,x_n$) if and only if $x_1,x_2,...,x_n$ are non-negative. Let x=($x_1,x_2,...,x_n$) and y=($y_1,y_2,...,y_n$) be two points with $x_1,x_2,...,x_n >0$ and $y_1,y_2,...,y_n >0$. Then it is easy to compute that

$$m\left(\frac{x}{y}\right) = \min_i \frac{x_i}{y_i}, M\left(\frac{x}{y}\right) = \max_i \frac{x_i}{y_i}$$

Thus we get the two new distances as follows.

$$\rho(x,y) = \ln\left\{ \max_{i,j} \frac{x_i y_i}{x_j y_i} \right\}, \quad d(x,y) = \ln\left(\max_{i,j}\left\{ \frac{x_i}{y_i}, \frac{y_j}{x_j} \right\} \right)$$

For general purposes, we give another definition of metric induced by a partial ordering.

Definition 6.4 We call the following function F-projective distance of the two points x, y:

$$\rho_F(x,y) = F\left(M\left(\frac{x}{y}\right), M\left(\frac{y}{x}\right), m\left(\frac{x}{y}\right), m\left(\frac{y}{x}\right) \right)$$

where F(a,b,c,d) is a function.

Now we describe a multi-objective optimisation scheme based on partial ordering technique. In design problems, we always use genotypes and phenotypes to denote two forms of a design solution. Genotypes will construct a space called *genospace*. We say one design is better than another design by saying that the phenotype of one design solution is better in some aspects than the second design solution. These better characteristics will correspond to some comparison principles of their genotypes. Of cause, there will be a number of such principles and they form a principle set. We will call the principle set *compatible* if we can say definitely one is better than the other. That is, if there are two genotypes *x* and *y*, and *x* is better than *y* while *y* is better than *x*, then *x* is equal to *y*. We will always consider compatible principle set.

Definition 6.5 Let the genospace be G and denote the comparison principle set by P . Define a partial ordering in G by principles in P . That is, $x \le y$ if and only if *y* is better than *x*.

Because the principle set is compatible, we see that the relation $x \le y$ is a partial ordering. Moreover, when the genospace is linear, we can obtain a cone by the principle set. This is possible when the genotypes are real vectors and they form a continuous region. Now we recall a famous lemma in set theory, the Zorn's Lemma. It tells that in a partial ordering space, if every totally ordering subset has an upper bound, then there is a maximal element. Interpreting this lemma in terms of design optimisation, we say that when we can get a best design from a series of design solutions, which have the property that if one design is better than the previous design, then we can get an optimal design in this direction. Thus we conclude with a proposition based on these definitions.

Proposition 6.1 There is an optimal design solution in any direction in a multi-objective design problem, provided that we can get a best design from a series of design solutions which have the property that one design is better than others.

This part of research is still on going and we expect to use more real design examples to verify these definitions and proposition.

7. System implementation and applications

Based on the theory described above, we have implemented a system based on 3D solid modelling techniques. Two integrated design studio environments have been integrated in this system. The platform is personal computers under windows 2000 or windows XP of Microsoft. The programming languages include Microsoft Visual C++ version 6.0, ACIS 3D Kernel, MatLab™ version 6.1 of MathWorks™, Inventor version 5 for additional solid modification. One of the features of the system is that all the sub-systems are fully compatible with commercial CAAD tools and systems, as well as rapid prototype facilities. A large number of object-oriented components of sophisticated surfaces and envelopes have been built using evolutionary techniques and partial ordering theory. Computational mechanisms have also been developed with which these basic data structures and components can be visualised, combined or split to allow new data structures or new forms to be derived using generative techniques. We are also exploring the possibility to scale up the applications with potentially thousands of solid objects with textual and spatial design details in our Global Virtual Design Studio powered by high performance computer and multiple VR projection facilities.

The first part of the system we have implemented is named as TowerDev. This is a fully controllable solid modelling environment with non-linear transform of existing solid models into design prototypes. The environment consists of an ACIS SAT viewer; Fly through viewer; Colouring and Rendering; and Programmable design routines for architectural envelopes. The main user interface of this part of the system is as follows:

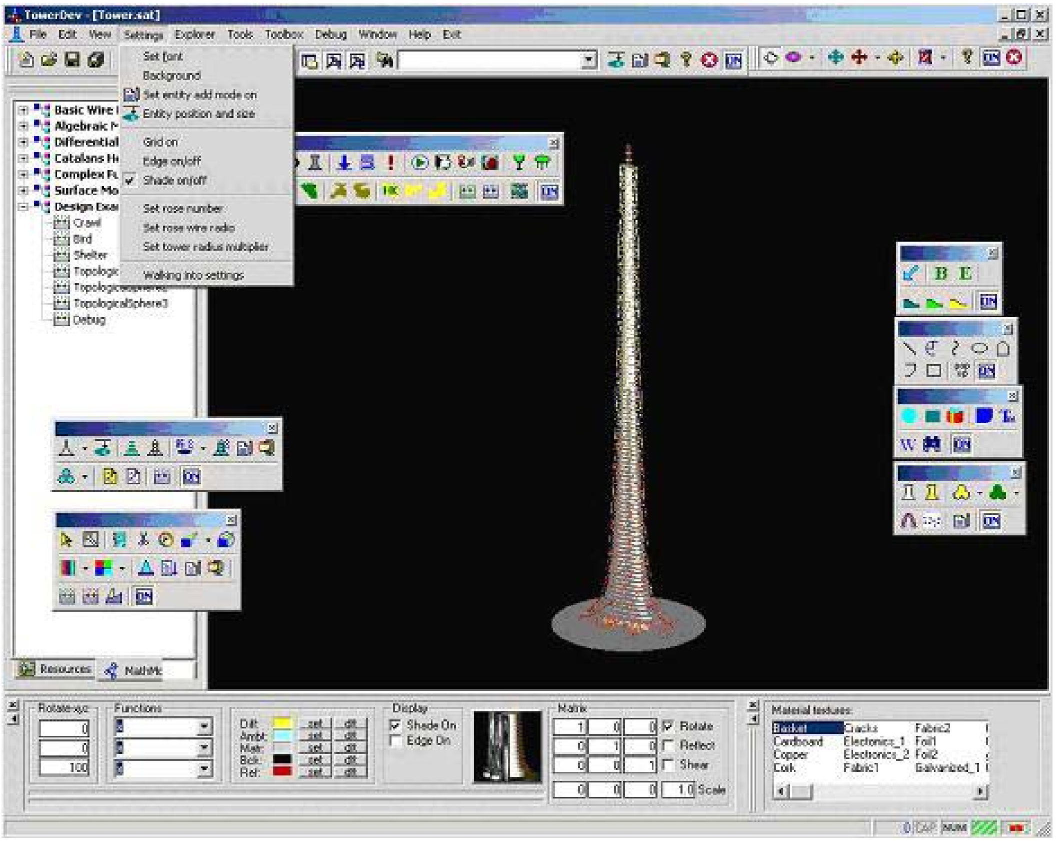

Fig 7.1 Graphical user interface of the system

Some design examples created by this system are illustrated as follows:

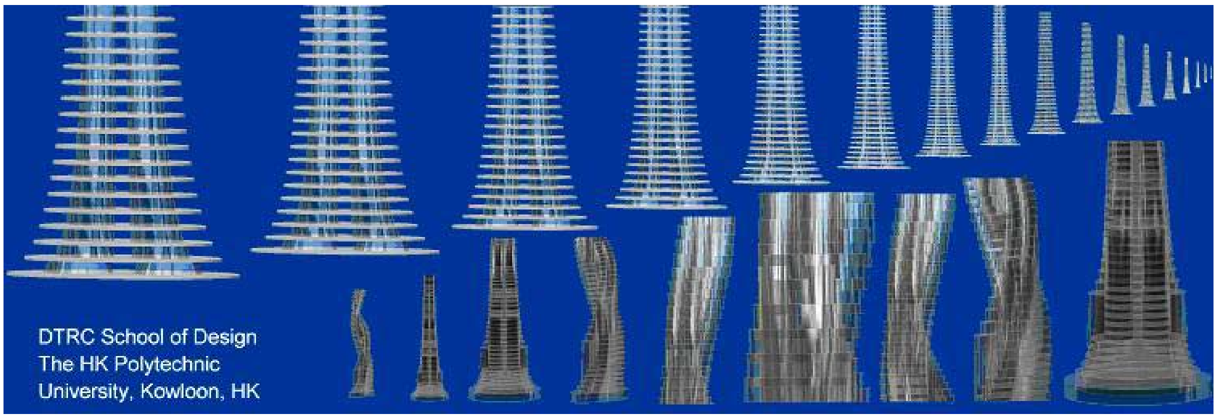

Fig 7.2 The variety of tower structures

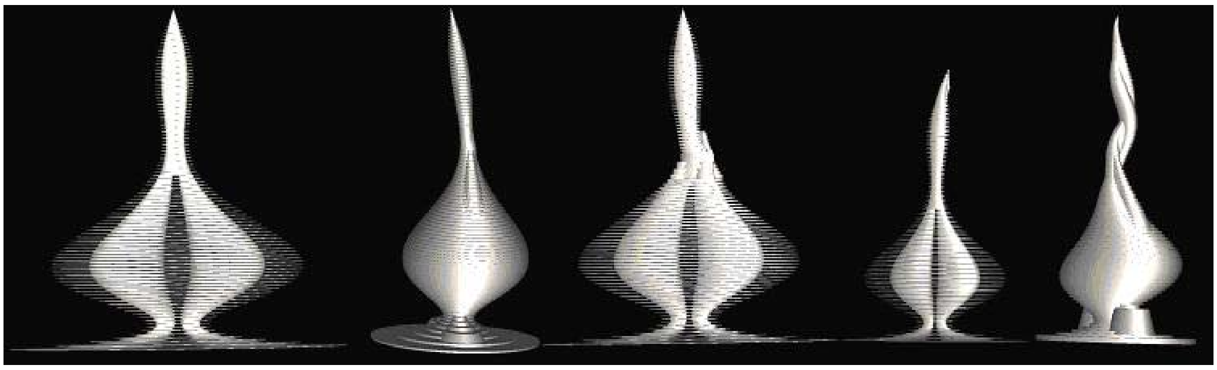

Fig 7.3 Models generated by functional envelopes

Acknowledgements

This research is partially sponsored by the Research Fellow Matching Fund Scheme 2001 (No. G.YY.34) of the Hong Kong Polytechnic University, the Young Scientist Award Project of Shandong Province of China, the Science and Technology Project of ShanDong Province, and the Natural Science Foundation of China.

References

[1] Frazer, J. H. (2001) Design Workstation on the Future. Proceedings of the Fourth International Conference of Computer-Aided Industrial Design and Conceptual Design (CAID & CD '2001), International Academic Publishers, Beijing, 2001; 17-23.

[2] Frazer, J. H. (1995) An Evolutionary Architecture. Architectural Association Publications, London, 1995.

[3] Frazer, J. H. (2000) Creative Design and the Generative Evolutionary Paradigm. In P. Bentley ed. Creativity and Design. In press. 2000.

[4] Gerd Fischer (editor), Mathematical Models, Friedr, Vieweg & John Verlagsgesellschaft mbH, Braunschweig, 1986.

[5] Tang, M. X. Knowledge-based design support and inductive learning. PhD Thesis, Department of Artificial Intelligence, University of Edinburgh, 1996.

[6] Tang, M. X. A knowledge-based architecture for intelligent design support. The Knowledge Engineering Review, 1997; 12(4): 387-406.

[7] Peter J. Bentley (1996), Generic evolutionary design of solid objects using a genetic algorithm. Thesis of doctor of philosophy, University of Huddersfield.

[8] Xiyu Liu, Tang M. X. and John H. Frazer (2002) Shape reconstruction by genetic algorithms and artificial neural networks. Proceedings of The 6th world Multi-conference on Systemics, Cybernetics and Informatics (SCI2002).

[9] Foley, J., van Dam, A., Feiner, S., Hughes, J. (1990), Computer Graphics Principles and Practice (Second edition), Addison-Wesley.

[10] Goldberg, D. E., (1989). Genetic Algorithms in Search, Optimisation & Machine Learning. Addison-Wesley.

[11] Frazer, J. H., Frazer, Julia, Liu Xiyu, Tang Ming Xi and Patrick Janssen, “Generative and Evolutionary Techniques for Building Envelope Design, Generative Art, Milan, 2002.