A Differentiable CGA Shape Procedural System

Victor Ceballos Inza
http://people.bath.ac.uk/vci20/
Department of Computer Science,
University of Bath

Yongliang Yang
http://www.yongliangyang.net
Department of Computer Science,
University of Bath

Tom Kelly
http://twak.org
School of Computing,
University of Leeds

Urban space design has been of continued interest for several centuries and procedural modelling is a tool that successfully addresses the challenges in the generation of cities. Procedural production systems are traditionally grammar-based, each one virtually describing a new programming language. Content generators using these tools need not only to have certain artistic sensitivity but also an ability to write computer programs.

There have been two recent directions that aim to ease the use of procedural systems. The first one, *inverse* procedural modelling, includes both systems that generate the production rules from square one [3, 5], and techniques that fit the parameters of existing grammars [6, 7]. The second one can be denoted *interactive* procedural modelling and introduces tools that leverage the troublesome process of parameter tuning of existing grammars, such as handles and local edits [1, 2].

We introduce in this paper the concept of *differentiable* procedural modelling, which enhances inverse modelling with tools from interactive methods. Procedural models have lots of confusing parameters which only the original programmer understands. We introduce an interactive tool which lets users perform direct modifications on the geometry of models, while it fits the parameters of a grammar to match those modifications. Formally, a *differentiable procedural system* is a parametrised production system [8], that is, a system composed of:

- An algebra of objects U, defined as $\langle U, +, -, F \leq \rangle$, where the set U is closed under the insertion $+$ and deletion $-$ operations and all transformations $f \in F$. An object $u \in U$ is said to occur in another object $v \in U$ if and only if there exists a transformation $f \in F$ such that $f(u) \leq v$.
- A set of production rules $R \subseteq U \times U$.
- A set of initial objects $I \subseteq U$.
- An interpretative mechanism, by which objects are generated by the repeated application of production rules to an initial object. Given an initial object $w \in U$, a production rule $u \to v$, and a variable assignment g, if there exists a transformation $f \in F$ such that $f(u) \leq w$, then the production rule can produce the new object:
$$[w - f(g(u))] + f(g(v))$$

such that:

- for each shape $w \in U$, each production rule $u \to v \in R$, each transformation $f \in F$ and each variable assignment g, we have that for every object $\tau \in U$ with
$$\tau \leq [w - f(g(u))] + f(g(v)) \in U$$
we can compute its rate of change (a.k.a., differentiate τ) with respect every variable assignment g^* applied in the generation of the shape
$$[w - f(g(u))] + f(g(v)) \in U$$

For a geometric interpretation of a differentiable procedural system, we develop a C++ implementation of CGA Shape grammars [4]. CGA Shape is a language for the procedural modelling of architecture. Our system works as follows. Given a set $\mathbf{p} = \{p_i\}_{i=1}^{n}$ of n input parameters, the procedural systems produces a geometry G. The user can change the parameters $\mathbf{p}$ to compute G, as in forward modelling. Interactively, the user can select with the mouse a vertex $\mathbf{v}$ in G and displace it to a different location $\mathbf{w}$ (inverse modelling), and our system automatically updates $\mathbf{p}$ to satisfy such a modification. We solve the minimization problem

$$\mathbf{p}^* = \arg\min_{\mathbf{p} \in \mathbb{R}^n} d(\mathbf{v}(\mathbf{p}), \mathbf{w})$$

for some distance measure d, which defaults to an Euclidean metric. Optionally, each parameter value p_i can be constrained within a plausible range $[p_i^{min}, p_i^{max}]$ or fixed to the current value if it corresponds to some feature of the geometry G that should stay constant. The (local) optimum is computed by a standard gradient-based non-linear function optimisation routine using a quasi-Newton strategy.

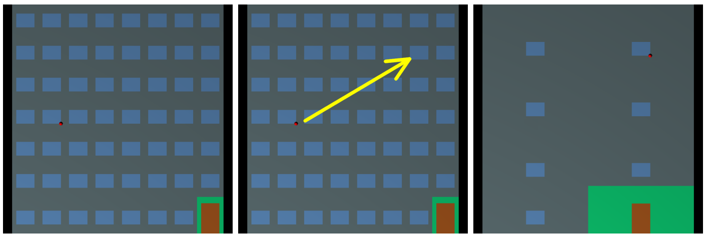

Figure 1: Illustration of the inverse modelling pipeline in our system. The user can select a vertex on the model and directly move it to a new location. The system updates the geometry according to the new parameters. Note that selected feature has kept its relative location (4th row, 2nd col).

The partial derivative of the vertex $\mathbf{v}$ with respect to each parameter p_i in the grammar is computed through the repeated application of the chain rule to the execution tree of G. G can be either be a initial shape $G \in I$ or be the result of a (or several) production rule. In the former case, G is a basic shape (polygon, polyhedra, circles, cylinders, etc.) and $\mathbf{v}$ can be expressed in terms of this shape's parameters. In the latter, G (or rather the subshape of G containing $\mathbf{v}$) is the result of applying some operator op to some other geometry G':

$$\mathbf{v} \in op(G', \{p_i\}_{i=1}^{r}) \subseteq G$$

where the subset $\{p_i\}_{i=1}^{r}$ of $\mathbf{p}$ contains exactly only the grammar parameters associated with op. Then the partial derivative of $\mathbf{v}$ is

$$\frac{\partial \mathbf{v}}{\partial p_i} = \sum_{j=1}^{m} \frac{\partial \mathbf{v}}{\partial \mathbf{u}_j} \frac{\partial \mathbf{u}_j}{\partial p_i} + \sum_{j=1}^{r} \frac{\partial \mathbf{v}}{\partial q_j} \frac{\partial q_j}{\partial p_i} \quad \text{for} \quad i \in \{1, 2, \ldots, n\}$$

where $\{\mathbf{u}_j\}_{j=1}^{m}$ are all the defining vertices in G', provided it is a polyhedron. If not, and G' is an analytical shape instead, these can be replaced by the defining shape parameters (radius, axis, etc.). These derivatives are computed manually using automatic differentiation.

Our system compares to [7] as both couple forward modelling with automatic inverse parameter fitting, but we remove the extra layer of urban indicators and allow direct modifications of the geometry. We do not use an approximation function for sampling and instead compute exact analytical derivatives.

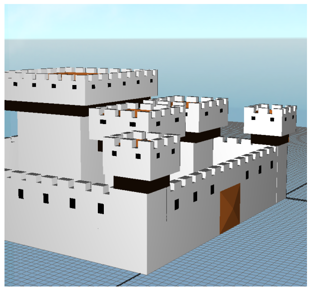

Figure 2: Sample of our results

[1] T. Kelly, P. Wonka, and P. Müller. Interactive Dimensioning of Parametric Models. In *Computer Graphics Forum*, pages 117–129, 2015

[2] M. Lipp, P. Wonka, and M. Wimmer. Interactive Visual Editing of Grammars for Procedural Architecture. In *ACM TOG*, page 102, 2008

[3] A. Martinovic and L. Van Gool. Bayesian Grammar Learning for Inverse Procedural Modeling. *IEEE CVPR*, pages 201–208, 2013

[4] P. Müller, P. Wonka, S. Haegler, et al. Procedural Modeling of Buildings. In *ACM SIGGRAPH 2006 Papers*, pages 614–623, 2006

[5] O. Šťava, B. Beneš, R. Měch, et al. Inverse Procedural Modeling by Automatic Generation of L-systems. In *Computer Graphics Forum*, pages 665–674, 2010

[6] J. O. Talton, Y. Lou, S. Lesser, et al. Metropolis Procedural Modeling. In *ACM TOG*, pages 11:1–11:14, 2011

[7] C. A. Vanegas, I. Garcia-Dorado, D. G. Aliaga, et al. Inverse design of urban procedural models. In *ACM TOG*, page 168, 2012

[8] P. Wonka, M. Wimmer, F. Sillion, et al. Instant Architecture. In *ACM SIGGRAPH 2003 Papers*, pages 669–677, 2003