SCRIPTING THE FUTURE

建筑数字化编程

Neil Leach & Philip F. Yuan
尼尔·里奇　袁 烽　编著

同济大学出版社
TONGJI UNIVERSITY PRESS

SCRIPTING THE FUTURE

建筑数字化编程

Neil Leach & Philip F. Yuan
尼尔·里奇　袁 烽　编著

同济大学出版社
TONGJI UNIVERSITY PRESS

支持单位
Supported by

同济大学建筑与城市规划学院教育部高密度人居环境实验室数字设计研究中心（DDRC）& 南加州大学中国学院（AAC）
Digital Design Research Center of Tongji University (DDRC) & The American Academy in China of USC (AAC)
（www.digitalfutureshanghai.com）

INTRODUCTION
介绍

运算设计在当今建筑设计的技术发展中扮演了重要的角色，新的软件和程序在建模、探索形式以及对整个设计和建造过程的控制上与过去的模拟技术相比有很大进步。但是运算设计所拥有的巨大潜力并不只在于对建筑设计实践的发展和改善，而是一种革命。正因为如此，新的算法技术的引入不是为了形式的建模，而是用编写代码的方式来创造形式。这种技术称作“脚本编程”，它让我们欣喜地看到了建筑设计领域中运算设计所蕴含的极大的潜力。

本书旨在通过对该领域最新发展动态的一览，展示新的计算技术，特别是编程对当代建筑实践的影响。本书的撰稿人涉及面甚广，从受数字化建造技术启发的商业公司，到对其不断研究的实验性工作室，再到材料商、工程师、学生以及各大院校的研究学者。但全书始终围绕着一条主线，那就是这些撰稿人在飞速发展的计算设计领域所作出的巨大贡献。

本书是这一系列中英文版的两本书中的一本，另一本是关于数字化技术与实际建造相结合的《建筑数字化建造》。这两本书是2011年在同济大学举办的“数字未来”展览的成果。这次展览是由同济大学建筑与城市规划学院和南加州大学中国学院共同主办，我们衷心地感谢惠普、《建筑实录》杂志、《时代建筑》杂志、McNeel公司、盖里科技、Permasteelisa公司、同济大学建筑设计研究院、同济大学城市与规划设计研究院、南加州大学中国学院等赞助单位，感谢你们促成了这次活动。同时，我们衷心感谢本书的撰稿人和所有协助本书编撰工作的人，包括杨璇、戴拉莫·汉切克罗夫特在资料收集整理中所做的工作；袁佳麟在全书编辑过程中所做的协助工作；平面设计团队——4aTEAM，以及戴祥萍、黄韧玲、尹文所做的翻译工作。

尼尔·里奇 & 袁烽

Computation has already played an important role in improving techniques of architectural design. New software programs for modeling and testing form, and for controlling the whole process of design and construction offer substantial advantages over previous analog techniques. But the real potential of computation lies, surely, not simply in improving the practice of architectural design, but in revolutionizing it. As such, the introduction of new algorithmic techniques not for modeling form, but for ‘breeding’ form through the manipulation of code — techniques that have become known as "scripting" — offer us a tantalizing glimpse of an even more radical potential role of computation within architectural design.

This volume seeks to offer an overview of the impact of new computational techniques — and scripting in particular — on contemporary architectural design, by providing a snapshot of the latest developments in the field. Contributors to this volume range from inspired, commercial firms to progressive, experimental firms, and from engineers to students and academic researchers. But the common thread throughout is an attempt to make a significant contribution to the rapidly developing field of computational design.

This volume is part of a two volume bi-lingual edition that also includes a volume on digital fabrication, *Fabricating the Future*. These volumes emerge out of the DigitalFUTURE conference and exhibition held at Tongji University in 2011 that were organized as a collaboration between the American Academy in China and the College of Architecture and Urban Planning of Tongji University. We are grateful to the sponsors who made these events possible: Hewlett Packard, *Architectural Record*, *Time+Architecture*, McNeel Associates, Gehry Technologies, Permasteelisa, Architectural Design and Research Institute of Tongji University, Shanghai Tongji Urban Planning and Design Institute, and the American Academy in China. We are also grateful to the contributors themselves as well as to all those who assisted in the production of this volume, including Sherry Yang, Deramore Hutchcroft, Crisie Yuan, the graphic design team — 4aTEAM, and the translators Dai Xiangping, Huang Renling and Yin Wen.

Neil Leach & Philip F. Yuan

CONTENTS 目录

PREFACE 绪论

URBAN FORMATIONS 城市生形

MATERIALIZATIONS 实践

CRITICAL COMMENTARY 评述

PATTERN LOGIC 模式逻辑

PROJECTS OVERVIEW 项目概述

PREFACE
绪论

Parametrics Explained
参数化释义

Neil Leach / University of Southern California
尼尔 • 里奇 / 南加州大学

如今数字设计飞速发展。尽管建筑师们使用计算机辅助设计（CAD）系统已有数十年之久，但也仅是最近几年两种独特有效的设计手段——参数化设计和算法设计才得以出现。得益于早期建筑研究者和程序员的实践操作，这些方法论才得到广泛的专业与学术认可。

这两种技术为建筑实践领域开拓了新的可能性。更重要的是，它们已在商业实践而非学术领域得到了飞速的发展和进步。自20世纪90年代以来，这些先进技术便与大量数字研究机构的出现保持同步，比如福斯特建筑事务所的“建模专家组”，盖里建筑事务所的“盖里科技”，奥雅纳的“先进几何单元”以及哈迪德建筑事务所的“编码”。这些内部数字研究机构是为了确保今天综合体建筑的设计和施工高效、准时和控制预算。我们也因此得以洞悉数字设计工具的发展演变，从20世纪90年代与科幻小说和虚拟现实相关，到21世纪初用其理解数字建构及材料特性，再到21世纪10年代它们成为综合体建筑中不可缺少的元素。

然而，这两种设计手段——参数化设计和算法设计通常会被混为一谈，有时甚至融合成另一个词“参数化主义”。在中国，这个情况特别突出，虽然“参数化”设计一词已经被几乎所有的大学接受并且用来指代算法设计，但是对于实际参数化软件的应用却是十分有限的。这不但导致了在特定形式的设计手法上的混淆，更在技术上错误地引导了它的发展。在西方，除非用到了参数化软件，很多使用算法设计的专家都尽力避免用“参数化”一词；但在中国，许多概念混淆的建筑评论家仍旧在用这一词。本文旨在澄清当前这一情形，为这两种独特的数字技术提供准确的定义，以便进行区分。这

Contemporary digital design practice is in a state of rapid evolution. While architects have employed computer-aided drafting (CAD) systems for decades, only recently have two distinct and potent design sensibilities — parametric and algorithmic design — emerged. Nurtured by early architectural researchers and programmers operating in practice, these methodologies are now gaining widespread professional and academic acceptance.

Together these two techniques are opening up a new field of possibilities for architectural practice. Most significantly, they have been developed and refined primarily in commercial practice and not in academia. Since the late 90s, these advances have coincided with the emergence of a number of digital research units within commercial practice, such as the Specialist Modelling Group at Foster and Partners, Gehry Technologies spun off of Gehry Partners, the Advanced Geometry Unit at Arup, and CODE at Zaha Hadid Architects. These in-house digital research units have been developed as a means of ensuring that the complex buildings of today are designed and constructed efficiently, on time and within budget. We can therefore discern an evolution within the development of digital design tools, from a period when they were associated with science fiction and virtual reality in the 1990s, through to a period when they began to be used to understand within the realm of digital tectonics to understand material behaviors in the 2000s, through to a moment when they have become almost indispensable in the production of complex buildings in the 2010s.

也表明了我们对“参数化主义”的态度，并且探讨是否把涌现视作为一种新的风格。①

参数化设计

参数化广泛应用于从数学到设计的各种学科。字面上来说，它指的是用规定的参数进行计算。但在现代设计的专业领域，参数化广义上指参数化建模软件的使用。与基于几何体的标准软件包不同，参数化软件将尺寸规格和参数与几何学相连接，通过局部的增量调整，影响整体组配。例如，随着曲线上某一点位置的变动，整个曲线也会重新排列。借助该手法进行的设计操作更具适应性，融合性与平滑性。因此它不仅可以用于建筑单体的建模，同时也可用于协同整个城市规划领域。当然，从建筑出现之时，就开始依靠“参数”，但用“参数化”这一术语来指“协同”设计的一种模式是近期才出现的。或许在这之后，用“协同”一词来指代我们现在所称的“参数化”软件会更合理些。

正如弗兰克·盖里、扎哈·哈迪德以及以形式操作为特征的其他建筑师的作品一样，参数化软件适用于曲线设计。然而，参数化软件本身并未开创形式。在参数化软件引入之前，有很多办公软件通过使用模拟技术，也可以建模。这些技术在重塑形式上效率更高，可以更好地控制设计流程。它们同时也为数字化建造过程提供更精确的信息。

然而，如果认为参数化设计仅仅涉及形式生成，那就大错特错了。相反，参数化设计给建筑师们提供了不同于传统手法的新的有效模式，以及协调施工流程的新方式（称为建筑信息模型）。正如数字化的项目，盖里科技为建筑业定制的CATIA建筑版本。该程序包的最大优势在于，允许施工团队在一个平台内沟通，对时间和成本控制得更有效。

算法设计

算法指使用程序技术解决设计问题。技术上来说，一种算法仅是一种指令。因此其与标准模拟设计流程的关系，类似其与数字设计流程的关系。然而，在数字化设计领域，算法尤指使用脚本语言，使设计师摆脱用户界面的束缚，直接通过操纵代码而非形式来进行设计。典型的算法设计通过计算机编程语言得以实现，如 RhinoScript, MEL (Maya内置语言), Visual Basic,或3dMaxScript。相反，由于编程很难，Generative Components及Grasshopper应用通过使用自动象形图表绕开代码。我们因此可以称之为“图形脚本形式”。算法设计挖掘计算机能力，像搜索引擎一样运作并执行可能会耗费大量时间。因此，算法设计可用于自我优化，以及其他超过标准设计限制的任务。②

However, these two design sensibilities — parametric and algorithmic design — are often confused, and sometimes collapsed into the single term, "parametricism". In China the situation is especially problematic, as the term "parametric design" has become adopted almost universally to refer to computational design, even though the use of actual parametric software is relatively limited. Not only does this often lead to the conflation of a method with a style, but it is highly misleading from a technical point of view. Although most leading exponents of computational design in the West now avoid the term "parametric" except when referring specifically to the use of parametric software, confusingly commentators in China still use the term. This article is an attempt to clarify the situation, and to offer some precise definitions of terms in order to differentiate these two quite distinct digital techniques. It is also an attempt to evaluate the term "parametricism" itself, and to question whether or not what we see emerging is a new "style" of architecture. ①

Parametric Design

Parametric is a term used in a variety of disciplines from mathematics through to design. Literally it means working within parameters of a defined range. Within the specific field of contemporary design, however, it refers broadly to the utilization of parametric modeling software. In contrast to standard software packages based on datum geometric objects, parametric software links dimensions and parameters to geometry thereby allowing for the incremental adjustment of a part which then affects the whole assembly. For example, as a point within a curve is repositioned the whole curve comes to realign itself. The operations that it facilitates are adaptation, blending and smoothing. It is therefore useful not only in modeling individual forms but also in the whole field of associative urban planning. Of course all architecture since the beginning of time has depended on the use of "parameters", but the actual use of the term "parametric" as an "associative" mode of designing is a recent development. In hindsight, perhaps, it would have made more sense to use the term "associative" to refer to what we now call "parametric" software.

Parametric software lends itself to curvilinear design, as in the work of Frank Gehry, Zaha Hadid, and other architects whose work is characterized by the manipulation of form. In itself, however, parametric software does not open up a new vocabulary of form. Such offices were modeling using analog techniques long before the

① 我特别感谢尼克·帕斯卡为本文提出的意见。本文中的某些章节是与他合作编写的，他为本文中论述的观点提供了宝贵的反馈意见。本文的早期版本曾发表于尼尔·里奇与徐卫国共同编著的《2010数字现实：学生建筑设计作品》，中国建筑工业出版社，2010。

I am deeply indebted to Nick Pisca for his advice on this article. Certain sections were co-written with him, and Nick has provided some invaluable feedback on the ideas expressed here. An early version of this essay was published in Neil Leach, Xu Weiguo (eds.), *Machinic Processes: Architects*, Beijing: China Architecture and Building Press, 2010.

参数化VS算法

现在普遍将“参数化”和“算法”这两个词合并，一定程度上是因为这两种技术都会产生相似的形式。例如，使用Processing或RhinoScript产生的算法模型通常为曲线形式，与使用参数化工具作出的模型类似。但这在一定程度上也是由于很少有人真正理解这两个术语的实际意义——至少那些不熟悉计算的人们如此。从某种角度上来说，“参数化”已成为以曲线为特点的数字设计的审美表达，进而也成为建筑新风格的一种便捷的表达。[1] 尽管制作的形式很相似，但生成这些模型的技术却大不相同。

事实上，若在现代学术环境中标出各种设计模式，会发现真正的参数软件的使用，如Digital Project实际上处于边缘地位。绝大多数会使用明确的建模技术，如Maya或3D Rhino，少部分会采用某种图像或编码自动化形式，仅仅非常小的一部分，可能就1%，会使用真正的参数化软件。Grasshopper, Processing以及RhinoScripting实际上是算法工具，不应称之为“参数化”。

尽管需要近义词或便于理解的术语来进行广义的概括，但这种发展未免让人觉得混乱，毫无疑问，也会引起进一步的困惑。在建筑文化中，采用的术语通常与其在文化中的用途含义并未有多大联系。实际上，纵观建筑史可以发现，一些术语，如“后现代主义”或“解构主义/解构”，在建筑领域指代建筑风格的应用，而这实际上是对其原有的文化含义的拙劣模仿。如今，参数化或参数化设计也面临同样的问题：尽管它确实是一种新的数字技术，促进了新的设计流程或设计方法，如今却被用于指代新的审美表达或建筑风格。

随着运算设计在建筑文化中变得越来越普遍，必须澄清相关的术语。在产生广泛混淆之前，使用参数化术语描述风格受到普遍支持。因此，让我们在这里澄清：算法技术是以代码的使用为基础；参数化技术是以形式处理为基础。因此，它们是截然不同的技术。

同时，通常算法技术与参数化技术共同使用。例如，使用某一算法技术生成原始形式，然后再通过参数化技术对形式进行处理。相反，在最初生成的形式通过参数化技术模型化后，算法技术可用于优化形式或用于设计过程结束阶段的其他操作。

然而，一种更深的担忧破坏了参数化和算法之间的明确区分，即设计工程中可视化的作用。许多建筑师并不是从基础开始学习编写代码，而是通过“私自篡写”屏幕中呈现的代码逻辑进行学习，其模型形式采用Maya或Rhino平台。换句话说，代码仅被用于提供“钥匙”以处理形式，且主要通过视觉进行理解。最近应用程序（例如Grasshopper）的使用进一步增强了对视觉的强调，通过使用自动

introduction of parametric software. However, these techniques are highly efficient for remodeling forms, and afford greater control in the design process. They also provide more precise information for digital fabrication processes.

It would be wrong, however, to assume that parametric design is concerned solely with form-making. On the contrary, parametric techniques afford the architect with new modes of efficiency compared to standard approaches and new ways of coordinating the construction process (called Building Information Modeling), as in the case of Digital Project, an architectural version of CATIA customized for the building industry by Gehry Technologies. The big advantage of such packages is that they allow the construction team to interface on a single platform, and afford a higher level of control in terms of monitoring the time and cost of construction.

Algorithmic Design

Algorithmic is a term that refers to the use of procedural techniques in solving design problems. Technically an algorithm is a simple instruction. It therefore relates as much to standard analog design processes, as it does to digital design processes. Within the field of digital design, however, it refers specifically to the use of scripting languages that allow the designer to step beyond the limitations of the user interface, and to design through the direct manipulation not of form but of code. Typically algorithmic design would be performed through computer programming languages like RhinoScript, MEL (Maya Embedded Language), Visual Basic, or 3dMaxScript. In contrast, due to the difficulty of programming, the applications Generative Components and Grasshopper bypass code with pictographic forms of automation. We might therefore describe them as forms of graphic scripting. Algorithmic design exploits the capacity of the computer to operate as a search engine, and perform tasks that would otherwise consume inordinate time. It therefore lends itself to optimization and other tasks beyond the limitations of standard design constraints. ②

Parametric versus Algorithmic

There is now a widespread practice of conflation the two terms, "parametric" and "algorithmic". This is partly due to the fact that the two techniques can end up producing similar forms. Algorithmic work generated using Processing or RhinoScript, for example, often has

② 应小心使用“优化”这一术语。通常我们不清楚最佳情况是怎样的，只是通过参数化来判断最优情况。若参数发生变化，最优情况的定义也将发生变化。我们指的是“改善的解决方案”而不是最优的解决方案。不管怎样，本文中“优化”一词不是指寻找最优方案的行为，而是指寻找它的过程。

The term "optimization" should be used with a degree of caution. It is often impossible to know what the optimum might be, and much depends on the parameters being used to judge that optimum. As those parameters vary, so the definition of the optimum will also vary. Instead of thinking in terms of optimized solutions we should perhaps refer to "improvements". Here, at any rate, the term "optimization" is used to refer not to the act of finding the optimum solution, but rather to the process of searching for it.

象形图表的形式绕开代码的使用。最终，我们不清楚究竟有多少建筑师在真实的算法框架下工作，而不是在视觉框架下进行简单的操作。

此外，可以认为参数化和算法操作之间有着某些相似之处，它们都是数值的容器，用户可通过输入改变数值。正如用户可以通过算法调整代码以产生不同结果，用户也可以通过参数化调整形式以产生不同的结果。这两种操作似乎都以参数的调整为基础。事实上，图形脚本处理技术的引入，例如Grasshopper，只会模糊参数化和算法设计间的区别。不仅如此，所有参数化设计都必然依赖于代码。换句话说，我们处于一种辩证的状态，代码和形式相互依赖。没有代码就没有形式，而没有形式也就没有代码。某种程度上说，算法设计和参数化设计只是同一枚硬币的两面。

但是，算法和参数化设计之间可以且必须做出明确区分。两者与参数联系的持续时间和永久性将其区分为不同的操作。此外，参数化设计很大程度上依赖于对形式的处理，这种形式可能在视觉上富有趣味，但往往只是表面的形式。例如，可以通过混合形状模型操作，以引人注目的视觉方式使形式变平滑（特别是从城市层面），但却无法考虑到任何与性能有关的有用信息，这类信息存在于建筑信息建模（BIM）模型内。

此外，应区分图形脚本处理技术（如Grasshopper）和非图案形式脚本处理。图形脚本处理应用范围相对有限。Grasshopper中没有足够强大的能与以文本为基础的脚本处理相媲美的代码树。事实上，图形代码没有文本编辑器，且在准确性、施工性、合理性和可扩展性上有明显的局限性。以代码为基础的脚本处理有着更高的控制性。尽管表面上相像，它们不仅在算法和参数化操作上，更在可视化脚本处理和纯粹的以代码为基础的算法操作上具有显著区别。

显然，参数化和算法技术能应对不同的设计挑战且都具有各自特点。建筑信息模型系统的广泛使用表明数字项目（一种参数化软件）对于建筑设计的未来具有重要作用。然而，有一种令人担忧的趋势，参数化建模（不亚于传统建模技术）主要被用于生成形式，特别是视觉上引人注意的形式。同样的，正如帕特里克·舒马赫认为的那样，参数化设计将有助于推广一种新的设计美学或风格。

然而，计算机时代孕育的不仅是一种新风格，而是全新的设计方法，我们将新的计算技术应用于进化的和涌现的系统中，建立并在现实中试验，使图解变成现实，现实变成图解。在这全新的视野下，形式变得不那么重要。我们应探索算法技术的潜在功能，并专注于更智能化和逻辑化的设计流程。逻辑便是新的形式。

curvilinear forms that are seemingly similar to work produced using parametric tools. But it is also partly due to the fact that as yet there is little real understanding of what the terms actually mean — at least on the part of those less familiar with the world of computation. To some extent the term, "parametric", has become a short hand way of bracketing much digital design that seems to be curvilinear in its aesthetic expression, thereby providing a convenient expression for a new style in architecture. The resultant forms may look similar, but the techniques for generating them are radically different.

In fact were we to chart the various modes of designing in the contemporary academic environment, we would probably find the use of genuine parametric softwares, such as Digital Project, to be fairly marginal. The vast majority might be using explicit modeling techniques, such as Maya or 3D Rhino, and a relative minority would be using graphic/coded automation of some form, but only a very small fraction — maybe 1% — would be using actual parametric software. Grasshopper, Processing and Rhino Scripting are actually algorithmic tools, and should not be referred to as "parametric".

Although there will always be a need for synonyms or accessible terms to describe broad approaches, this development is a little disturbing, and will lead no doubt to further confusion. Often within architectural culture terms have been adopted that have little relevance to their use in culture at large. Indeed we can look through the history of architecture, and find several terms, such as "Postmodernism" or "Deconstructivism/Deconstruction", adopted within an architectural context to refer to architectural styles in a way that has made them reduced parodies of their original cultural meanings. Parametrics or parametric design now seems to be suffering a similar fate: although it is actually a new digital technique that fosters a new process or methodology of design, it has now been adopted to refer to a new aesthetic expression or style of architecture.

As computation becomes increasingly prevalent within architectural culture, some effort must therefore be made to clarify the terms, before the confusion becomes so widespread that the use of the term parametric to describe a style is sanctioned through sheer popularity. So let us state here quite clearly: Algorithmic techniques are based on the use of code. Parametric techniques are based on the manipulation of form. They are therefore quite distinct techniques.

At the same time algorithmic techniques are often used in association with parametric techniques. We might point, for example, to the use of certain algorithmic techniques to generate the initial form that is subsequently manipulated using parametric techniques. Conversely, algorithmic techniques can be used for optimization and other operations at the other end of the design process, once the initial generated form has been modeled through parametric techniques.

There is, however, a deeper concern that destabilizes this neat distinction between the parametric and the algorithmic, and this is the role of visualization in the process of design. Many architects do not learn to write code from scratch, but learn by "hacking into" the logic of the code displayed in boxes on the screen, as they model forms using platforms such as Maya or Rhino. In other words, the code is used merely to provide the "key" to the manipulation of form, and can be understood primarily through the medium of the visual. This emphasis on the visual is heightened further with the introduction of recent applications — such as Grasshopper — that bypass code by using pictographic forms of automation. In the end it is not so clear how many architects are working within a truly algorithmic framework, as opposed to simply operating within a visual framework.

Further, it could be argued that parametric and algorithmic operations share certain similarities, in that both are containers in which values can change based on user input. Just as one can adjust the code algorithmically to generate different outcomes, so one can adjust the form parametrically to generate different outcomes. Both operations appear to be based on the adjustment of parameters. Indeed the introduction of graphic scripting techniques, such as Grasshopper, serves only to blur the distinction between parametric and algorithmic design. Moreover all parametric design relies necessarily on code. In other words we find ourselves in a dialectical situation where code and form rely upon one another. There can be no form without code, and often no code without form. To some extent, then, algorithmic design and parametric design are merely two sides of the same coin.

Yet there are clear distinctions that can and must be made between the algorithmic and parametric design. Importantly, it is the duration and permanency of their connections to parameters that distinguish them as operations. Further, parametric design depends greatly on a manipulation of form that might appear visually interesting, but is often superficial. The blend-shape modeling operation, for example, can be used to smooth out form in a seductive visual way — especially at an urban level — but cannot take account of any useful performance related information embedded in a Building Information Modeling (BIM) model.

Moreover, a distinction should made between graphic scripting techniques, such as Grasshopper, and non-pictorial forms of scripting. Graphic scripting is relatively limited in its scope. Seldom are there any code trees to be found in Grasshopper robust enough to compete with the average text based scripting. Indeed graphic codes have no text editor, and have severe limitations in terms of accuracy, constructability, rationalization and scalability. Code based scripting enjoys far greater levels of control. Despite their apparent similarities, then, there are important distinctions to be made not only between algorithmic and parametric operations, but also between visual scripting and purely code-based algorithmic operations.

Clearly parametric and algorithmic techniques respond to different design challenges, and both have an important role to play. Moreover, the increasing use of Building Information Modeling systems suggests that Digital Project — as a parametric software — has much to contribute to the future of architectural design. And yet there is a worrying tendency, it would seem, for parametric modeling — no less than explicit modeling techniques — to be used primarily for the generation of form, and visually seductive form in particular. As such, parametric design threatens to help promote, as Patrik Schumacher argues, a new design aesthetic or style.

Surely what the world of computation promises is not merely a new style, but a radically new way of approaching design, where we embed new computational techniques into evolutionary and emergent systems, and where we breed systems and test them out in real time, so that the diagram becomes the reality and reality is the diagram. Form should be seen as largely irrelevant within this new horizon. Instead we should be exploring the potential of algorithmic techniques, and focus on more intelligent and logical design processes. Logic should be the new form.

参考文献 / References :

[1] Patrik Schumacher, "Parametricism: A New Global Style for Architecture and Urban Design" in Neil Leach (ed.), Digital Cities, *Architectural Design*, Vol. 79, No. 4, July/August 2009, pp. 14-23; this article is reprinted in this volume.

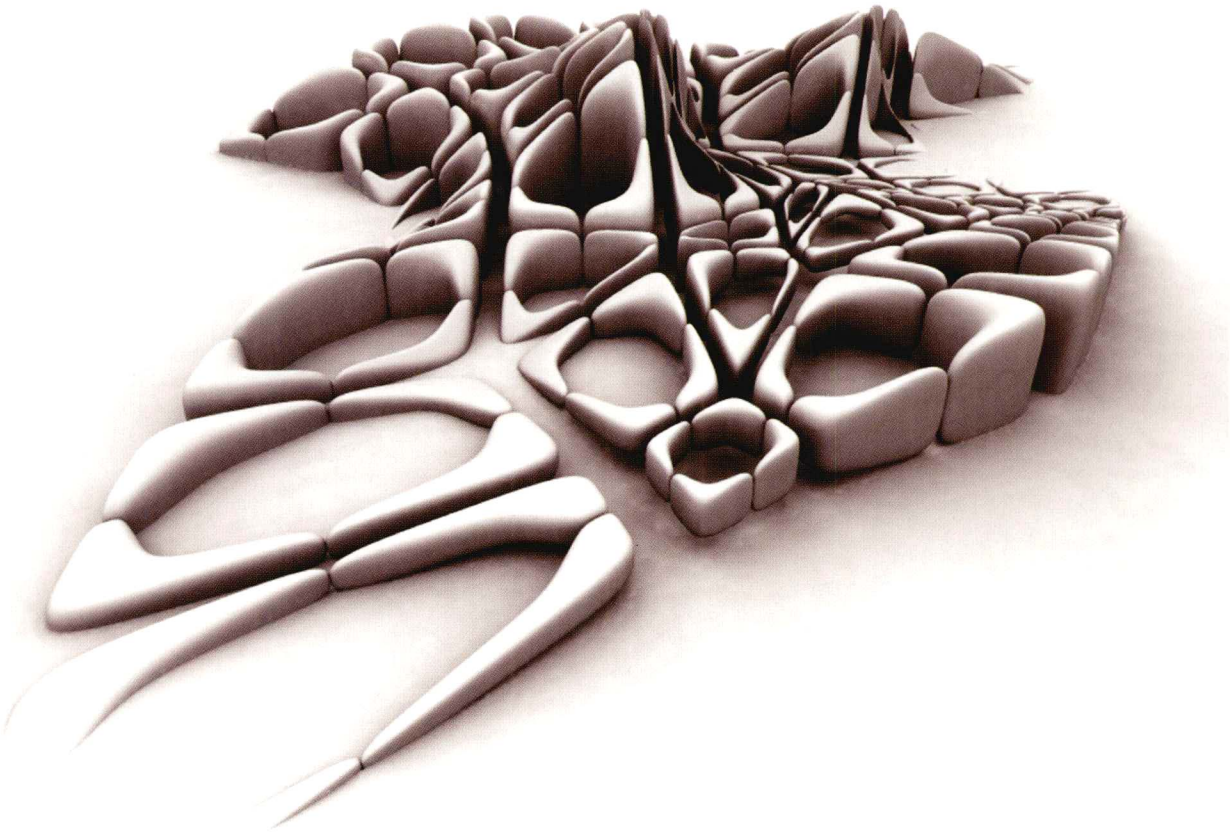
URBAN FORMATIONS
城市生形

Patrik Schumacher / Zaha Hadid Architects
帕特里克·舒马赫 / 扎哈·哈迪德建筑事务所

Neil Leach / University of Southern California
尼尔·里奇 / 南加州大学

Michael Weinstock / AA EmTech
迈克尔·维斯托克 / 涌现技术研究室，伦敦建筑联盟学院

Tom Verebes / The University of Hong Kong
汤姆·沃勒比 / 香港大学



扎哈·哈迪德建筑事务所，土耳其伊斯坦布尔Kartal-Pendik总体规划，2006
肌理研究：城市肌理包括塔楼和参数街区。图片展示了参数街区类型的形态范围。街区分成四个区域后，即可引入次级步行系统。某些网络交叉点处的街区系统与塔楼系统相似：每个街区均为一个区域，形成环绕网络交叉点的虚拟塔楼。
Zaha Hadid Archiects, Kartal-Pendik Masterplan, Istanbul, Turkey, 2006
Fabric study. The urban fabric comprises both cross towers and perimeter blocks. The image shows the morphological range of the perimeter block type. Blocks are split into four quadrants allowing for a secondary, pedestrian path system. At certain network crossing points the block system is assimilated to the tower system: each block sponsors one of the quadrants to form a pseudo-tower around a network crossing point.