Backcasting and a New Way of Command in Computational Design

*Reinhard Koenig, *[1,2,3] Gerhard Schmitt[1,3]*

*[*1] Corresponding Author: Reinhard Koenig,*
ETH Zurich, Department Architecture, Chair of Information Architecture
[1] email: reinhard.koenig@arch.ethz.ch, web: http://www.ia.arch.ethz.ch/koenig/
[1] ETH Zurich, Future Cities Laboratory, Singapore-ETH Centre, Singapore
[2] Junior-Professorship for Computational Architecture, Bauhaus-University Weimar, Germany
[3] ETH Zurich, Department Architecture, Chair of Information Architecture, Switzerland

Abstract: *It's not uncommon that analysis and simulation methods are used mainly to evaluate finished designs and to proof their quality. Whereas the potential of such methods is to lead or control a design process from the beginning on. Therefore, we introduce a design method that move away from a "what-if" forecasting philosophy and increase the focus on backcasting approaches. We use the power of computation by combining sophisticated methods to generate design with analysis methods to close the gap between analysis and synthesis of designs.*

For the development of a future-oriented computational design support we need to be aware of the human designer's role. A productive combination of the excellence of human cognition with the power of modern computing technology is needed. We call this approach "cognitive design computing". The computational part aim to mimic the way a designer's brain works by combining state-of-the-art optimization and machine learning approaches with available simulation methods. The cognition part respects the complex nature of design problems by the provision of models for human-computation interaction. This means that a design problem is distributed between computer and designer.

In the context of the conference slogan "back to command", we ask how we may imagine the command over a cognitive design computing system. We expect that designers will need to let go control of some parts of the design process to machines, but in exchange they will get a new powerful command on complex computing processes. This means that designers have to explore the potentials of their role as commanders of partially automated design processes.

In this contribution we describe an approach for the development of a future cognitive design computing system with the focus on urban design issues. The aim of this system is to enable an urban planner to treat a planning problem as a backcasting problem by defining what performance a design solution should achieve and to automatically query or generate a set of best possible solutions. This kind of computational planning process offers proof that the designer meets the original explicitly defined design requirements.

A key way in which digital tools can support designers is by generating design proposals. Evolutionary multi-criteria optimization methods allow us to explore a multi-dimensional design space and provide a basis for the designer to evaluate contradicting requirements: a task urban planners are faced with frequently. The vision for a cognitive design computing system is to enable an urban planner to treat a planning problem as a backcasting problem by defining what performance a design solution should achieve and to automatically query or synthesize a set of best possible solutions.

In another part we reflect why designers will give more and more control to machines. We investigate first approaches learn how designers use computational design support systems in combination with manual design strategies to deal with urban design problems by employing machine learning methods. By observing how designers work, it is possible to derive more complex artificial solution strategies that can help computers make better suggestions in the future.

Keywords: *Cognitive design computing, backcasting, machine learning, evolutionary optimization, design synthesis*

DOI: *10.3311/CAADence.1692*

1. INTRODUCTION

New types of computing, such as cognitive computing, are extending the application of IT into new areas. A general definition of cognitive computing is "the simulation of human thought processes in a computerized model" (Rouse, 2014). IBM's Watson computing initiative and the associated programs, represent an important development in this direction. A computational device as an opponent in a game of chess or a computer that triumphs on Jeopardy is just the beginning, and many other applications will follow (Kelly & Hamm, 2013). The latest success of a machine was DeepMind's program AlphaGo, which beat a human professional player in the ancient game of Go (Gibney, 2016). The approach we present in this contribution focuses on the application of cognitive computing to the domain of urban design. In our context, urban design means the arrangement and proportioning of spatial elements such as streets, open spaces and buildings taking into consideration functional aspects like accessibility, visual qualities, or solar radiation. Our ultimate objective for this cognitive design computing system, as we call it, is to develop a program that is able to make urban designs

that are comparable to or even better than human designs. For this, the cognitive design computing system needs to be able to learn from existing designs as well as from human design strategies. The way this system interacts with designers is therefore a crucial aspect. To be able to evaluate the performance of a design, we need various evaluation methods. Known urban analysis and simulations methods are one option, systematically collected ratings by humans another.

Architecture and urban design have always been excellent application areas for artificial intelligence and cognitive computing, but the small relative and absolute numbers of researchers in architecture made the advances appear less significant than they actually were. Design applications of artificial intelligence methods and techniques were introduced into education as early as the 1970s and 1980s in the United States (Mitchell, 1977) and later in Europe. Architecture is an interesting application area, because it involves a combination of structured input that can be produced with rule-based systems and the appraisal of past experiences and expectations of the future. This mix of requirements corresponds almost exactly to the computational tools already used: structured input and constraints, e.g. as defined by city authorities; historical data and information, which can serve as the basis for future design decisions; and user requirements that come in very different shapes and sizes and representations.

Urban design is an even more interesting application area of cognitive computing, as the amount of structured information and rules is relatively small compared to architecture, but the amount of decisions that can be derived from input from citizens, transportation needs, and external requirements is much higher than similar information for individual buildings. To achieve this, we need better ways of collecting and mining opinions, proposals, and requests that can be represented as data.

Cognitive design computing can be understood as a combination of the above: from architectural design, it draws on the very efficient abstraction methods and deep knowledge of materials, climates, and people's use of habitats that go back thousands of years. From urban design, it draws on the necessity to provide for large numbers of people that do not necessarily live in the urban system, but which rely on its infrastructure and central functions. The advent of big data is of relevance for both cases as mining big data for patterns and individual preferences has the potential to make urban design computing systems more and more powerful.

In the following sections we introduce the framework of our cognitive design computing system and its four main parts, as shown in Figure 1: data analysis, user interaction, learning, and geometry. These we then examine in detail in the sections that follow. Data analysis focuses on the indexing of geometries or of spatial configurations in general. A prerequisite for indexing is that we can distinguish geometries. For this purpose, we introduce a method of individually 'fingerprinting' spatial configurations. The section on user interaction and learning describes how the cognitive skills of the designer are involved and how the computer system can learn problem solving strategies by observing the user's actions. The geometry section introduces methods for synthesizing spatial configurations, which are in turn used as input for the data analysis. We focus on an automatic synthesis procedure to show first examples for the generation of pareto-optimal design solutions. We demonstrate the automatic synthesis procedure using the example of a concrete planning project. Finally, we conclude with a reflection of the developed system and outline next steps for its further development in the outlook section.

2. COGNITIVE DESIGN COMPUTING FRAMEWORK

The central objective of the cognitive design computing system is to serve as a planning support tool that can lead the designer towards better solutions or suggests new, useful alternative solutions. The technical realization can be either as a plug-in for existing systems or separate tool. In both cases a separate user interface is needed to allow the designer to design interactively using the system. The main elements of the cognitive design computing framework are illustrated in and explained in the following.

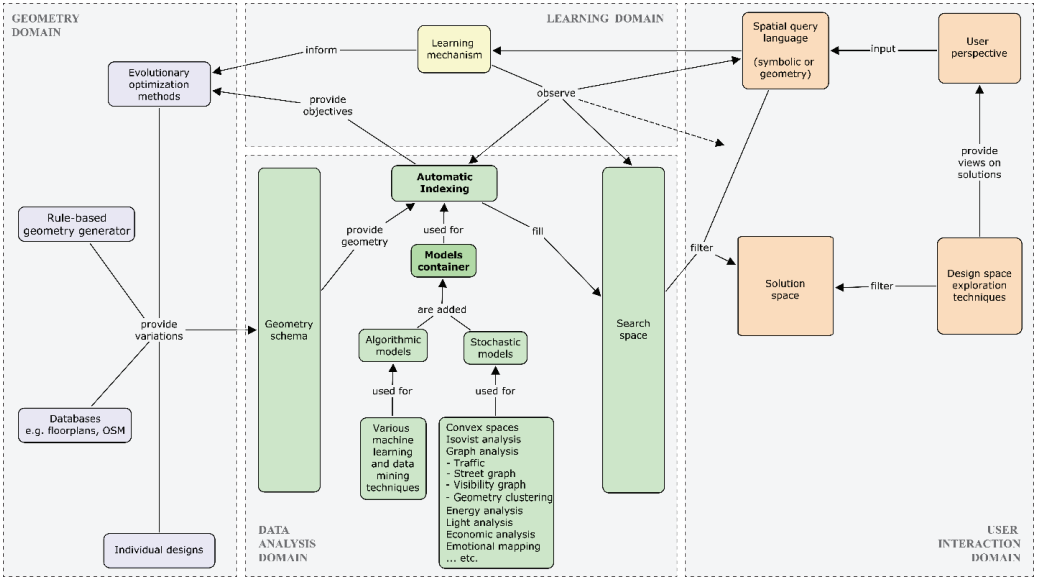


Figure 1:
Cognitive Design Computing Framework. The main domains are data analysis, user interaction, learning, and geometry.

Unlike approaches that employ urban data to exclusively analyze existing situations, the intention of cognitive design computing is to transcend the retrospective view by integrating data via a model container into the urban design and planning process (). The main technique for doing this is the model container, which is shown in the data analysis domain in . The model container can hold all models that describe relationships between the built environment and any kind of data. Rather than undertaking a systematic analysis of which models or data would be necessary for a comprehensive, rational planning process, we take an opportunistic approach and adapt a concept described by Maruani and Amit-Cohen (2007, p. 5), using models for which data is available or that are promising for certain planning problems.

3. DATA ANALYSIS

The data analysis part of the cognitive design computing framework aims first to distinguish designs by analyzing their geometry, and second to add as many indexes to the geometries. The corresponding problems are to find a generalizable way to create individual fingerprints for any kind of geometry, and to be able to aggregate results from various kinds of analysis (from the models container) to indexes.

We start with the issue of how to compare different designs based on their geometric representation. Existing methods can be divided into two main groups according to the particular kinds of compared data: either they compare shape determining rules instead of a shape itself (Stiny & Mitchell, 1978) or they compare characteristic values computed as shape features of a design (Derix & Jagannath, 2014; Dillenburger, 2010) and topology informed labels (Langenhan, Weber, Petzold, & Dengel, 2011). A more general method for characterizing or labelling designs based on their purely geometric representation was introduced by Standfest (2014). He proposed a Deep Learning method for unlabeled 3D polygon meshes. The resulting characterization of a design can be understood as its fingerprint. This method can be considered as algorithmic modeling and is part of an observable trend towards minimizing the amount of semantic information needed for state-of-the-art data analysis. Because the method is domain independent, it can be applied at various scales, e.g. to evaluate apartment plans, buildings

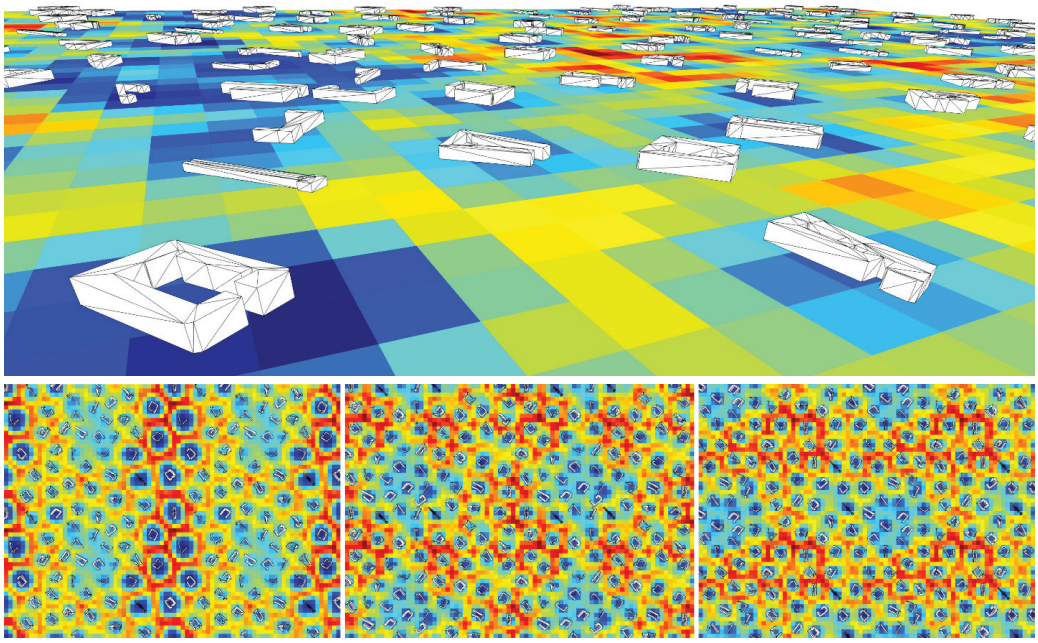


Figure 2: Result of the experiment conducted on clustering 48 randomly chosen building blocks according to the latent semantics of the unlabeled mesh geometry (Standfest, 2014). The building blocks are preprocessed with a Delaunay triangulation for each plane. Bottom row, left: Domain maps of level 1 (small mesh face neighborhoods consisting of 4 triangles), middle: level 2 (slightly bigger mesh face neighborhoods, consisting of 9 triangles) and right: level 3 (biggest mesh face neighborhoods consisting of 22 triangles).

(Figure 2), facades, streets or whole neighborhoods. For a shallow learning approach, other spatial entities can be used instead of polygon meshes.

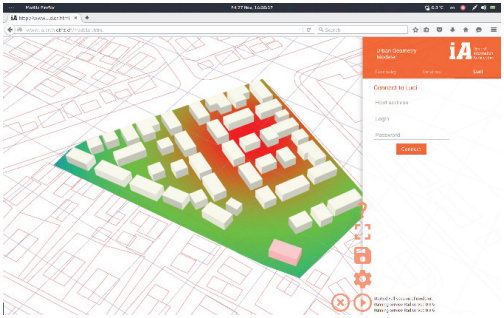
Such similar sized feature vectors, also referred to as fingerprints, are important for data analysis, especially in the context of cognitive design computing. Firstly, they can be used to distinguish geometries to ensure that only significantly different ones are added to the search space. And secondly, it makes it possible to correlate geometry with empirical observations, sensor data, or computed measures from stochastic models.

We applied the method by Standfest (2014) to create the fingerprints of 48 building volumes randomly chosen from the district of Zürich Altstetten and provided by the city of Zurich. After labelling the buildings, they are clustered using a Self-Organizing-Map (Figure 2). Despite the limited volume of the data set, the resulting maps of different abstraction levels show significant clustering and topologically correct alignment of the evaluated building blocks. Since the approach is strictly data driven, the characterization of design alternatives may differ from those of a designer. The example application shown in Figure 2 illustrates how different unlabeled polygon meshes can be aligned according to latent semantics.

4. USER INTERACTION AND LEARNING

In this section we combine user interaction and learning methods. First we aim to collect empirical data that can be used for indexing geometry, and second we observe how a human designer solves a design task and learn from it to derive an artificial design strategy. The long-term objective is to combine the strengths of human observation, cognition, experience and local knowledge into our system to improve the planning, design, management and transformation of buildings and cities.

Based on the models container described in section 3, we have various ways to measure the qualities of an existing or a new urban design, depending on social, cultural, and functional contexts. For instance, one could calculate the level of street noise, air pollution, or solar exposure. With this



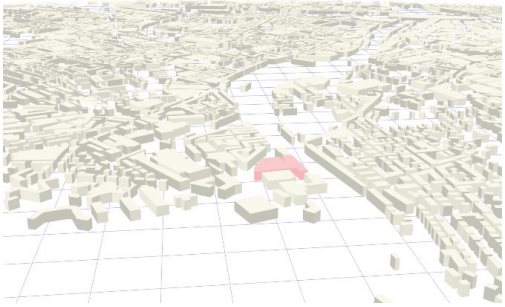


Figure 3:
Web user interface for citizen design that allows geometry to be created and manipulated (developed by Artem Chirkin for his PhD). Various urban simulation and analysis tools can be run on a webserver and the results can be visualized (Figures by Artem Chirkin).

in mind, we assume the designer always pursues a number of goals in the form of criteria and constraints when developing a design. If a machine could know the formal descriptions of the criteria and their importance weighting, it could also optimize a design accordingly. The quality of the solution then depends on the quantity and quality (sensitiveness) of such design criteria, as well as on an estimation of the user's goals for these criteria. The challenge in implementing our learning mechanism is to develop an algorithm that estimates the user's preferences with regard to the various design performance measures.

To provide an adequate user interface for human-computer interaction, we developed an initial prototype that make use of current web-based technologies to render urban designs and simulation results via a web browser (Figure 3). It facilitates visualizing, editing, creating, and evaluating spatial configurations at various scales. Various urban simulation and analysis tools can be run via a webserver using LUCI (Treyer, Klein, König, & Meixner, 2015) as middleware and the results can be visualized on the website. In addition, the web user interface can be used to present a design problem and observe the strategy a designer applies to find a spatial solution. These can then be used as input for a learning mechanism with the aim of applying it independently to new similar design tasks.

Through the learning domain we aim to implement a design routine that, on the one hand, proposes design alternatives to a planner and, on the other, obtains feedback in the form of the selection of a design variant to proceed with, thereby helping the system learn and adapt to the user's needs.

5. GEOMETRY

Beside using existing urban designs or manually creating them, another crucial part of the cognitive design computing framework is to automatically synthesize geometry. According to Weber, Müller, Wonka, and Gross (2009), the synthesis of urban structures consists of a sequence of several processes: the creation of a road network, the definition of land use and parcelling, and building placement. Systems have been developed for the procedural creation of road networks based on L-systems (Parish & Müller, 2001). In particular, the system CityEngine by ESRI facilitates the three-dimensional, rule-based modelling of cities and urban structures to the level of building details (Gool et al., 2006; Weber et al., 2009). In all these examples, the rules for the creation of an urban design solution have to be specified a priori in detail. The rules of generative or procedural algorithms are also very technical, abstract and not related to a planning problem. More importantly, they are not combined seamlessly with evaluation models and optimization methods. With these methods we therefore "have a model that can generate designs but has no means of establishing whether those designs are any good" (Radford & Gero, 1988, p. 20).

To achieve more advanced and more meaningful geometry synthesis, we therefore need to find a representation that is able to create realistic geometry for a design and can incorporate a lot of performance measures (objective functions) that can be defined by a designer. This information should make it possible for the synthesis system to generate a correspondingly large amount of possible design solutions without needing to then

analyze if objectives contradict or not. To this end, we introduce a primary method for synthesizing geometry using Evolutionary Multi-criteria Optimization (EMO) and show how this method is applied in a synthesis case study.

5.1. Evolutionary Multi-criteria Optimization

The basic technique we use for synthesizing geometry is evolutionary algorithms (EA) due to their flexibility with regard to problem representation as well as their robustness. This allows us to flexibly experiment with how we technically encode a design problem in the knowledge that the EA still work in an acceptable way even if we have a poor technical implementation. EA can be applied on various scales for layout design (Koenig & Knecht, 2014), building volume arrangement (Koenig, 2015b), urban district planning (Knecht & Koenig, 2012), or network development (Koenig, Treyer, & Schmitt, 2013; Schaffranek & Vasku, 2013). The EA may be supplemented by a number of local search strategies in order to optimize its calculation speed (Koenig & Schneider, 2012).

When we extend EA to include more sophisticated selection mechanisms that are able to consider more than one objective function for the evaluation of design solutions, we speak of Evolutionary Multi-criteria Optimization (Deb, 2001). For our design synthesis prototype we developed an individual evolutionary strategy in combination with a selection mechanism using the HypE algorithm (Bader & Zitzler, 2011) from the PISA framework (Zitzler & Thiele, 1999). This allows us to filter the non-dominated solutions out of all generated solutions, especially if we have to deal with a variable set of contradicting and non-contradicting criteria. During the computer-supported design process, planners obtain immediate feedback in the form of a set of design solutions that fulfill the formulated design requirements as well as possible. The presented system for synthesizing designs offers the possibility to experiment with various restrictions and objectives for a design project. This is an important feature since the definition of a design problem can be considered as a main step towards its solution (Rittel & Webber, 1973).

5.2. Synthesis Case Study

We assessed the applicability of the EMO method for geometry synthesis using an example scenario in Singapore. To demonstrate how our approach works in an existing urban context, we chose a defined area and assumed it needed to be completely re-planned (Figure 5, a). The choice of this example in Asia reflects an urgent need for fast and comprehensive planning systems. Necessary data on the street network was taken from Open Street Map, and information about neighboring built structures in 3D was available from the Future Cities Laboratory of the Singapore ETH Centre.

We apply the design synthesis methods for creating road networks with defined centrality characteristics, such as integration or choice of specific locations. We used these to define a location with high centrality for a new central business district, and a separate location with quite low traffic for a new residential area. Both requirements cannot be fully fulfilled, since they contradict each other where the locations adjoin. Here we need the ability of the EMO to find pareto-optimal solutions for contradicting problems. A set of these best compromise street networks is shown in (Figure 4c). Inside the blocks of the road networks we generate building layouts with defined densities, taking into account specific properties of the open space qualities measured by Isovist fields. Again these criteria may contradict each other. A set of generated pareto-optimal building layouts is shown in (Figure 4b). We illustrate how a user can interact with the developed prototype system (Figure 4) and how it can be used to help develop an urban planning proposal in a step-by-step approach (Figure 5).

The planning process starts with the empty planning area defining the border for placing new street segments, and the starting street segments from which the street network is grown. The starting segments are taken from the existing network where it intersects with the planning area (Figure 5, a). Initially the user has to execute the EMO first for the street layouts and later for the building placements by specifying the respective properties on the right-hand section of the software prototype window shown in Figure 4d.

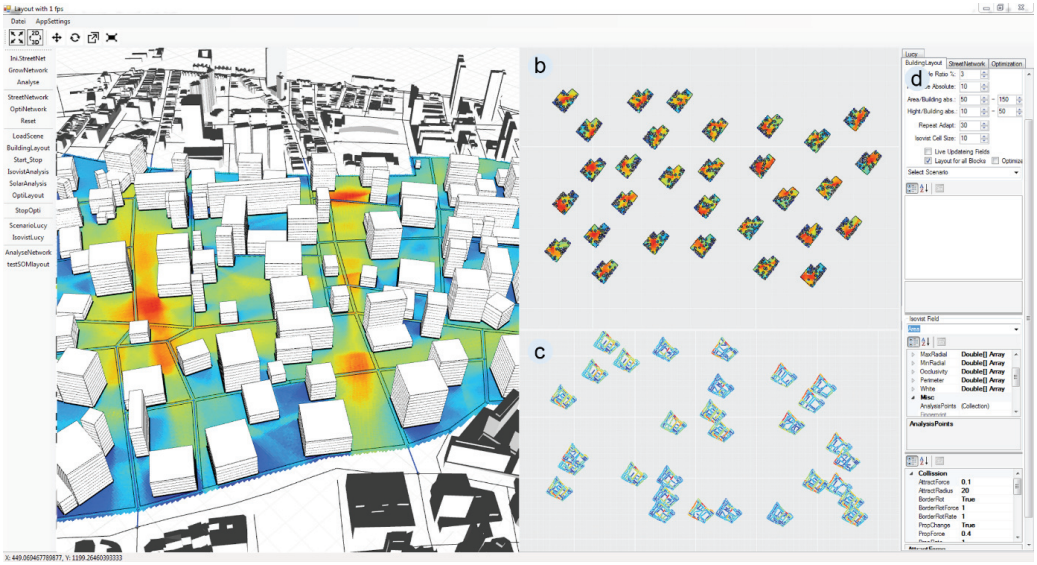


Figure 4:
Software prototype showing the main areas of the user interface: (a) 3D view combining one solution out of each archive; Design solutions of the archives for (b) buildings layouts and (c) street networks; and (d) fields for inputting the size of population, number of generations, etc.
A demonstration video of the prototype is available at http://cplan-group.net/demo/

The user can, for example, select the size of the population, the number of generations to calculate optimal layouts, and the size of the archive to store the solutions. The user interface shown in Figure 4 is structured in three main areas for visualizing the generated spatial configurations (6a-c). Figure 4b and 6c show the archives of best variants for the building layouts (6b) and street networks (6c) generated so far, and (6a) presents a 3D view that shows the configurations selected by a user out of the archives.
The centrality analysis can be run for the new network connected to the existing network in a user-defined radius around the planning site. They are combined with each other and the environment's geometry. Based on our representation of a design by the chromosome structure of the EA, our software prototype makes it possible to move, rotate and scale individual objects (street segments and building volumes) during the planning and optimization process. This is made possible by a specially-developed mechanism that sends information on the changed geometry to its numeric representation in the chromosome. This is a very important feature of the system, since it allows a designer to modify selected urban design solutions according to their individual needs during the optimization process.
Corresponding view control functions for zooming, panning and rotating the view are available for each of the views of the software prototype (Figure 4). After several iteration steps, street graphs and building layouts appropriate to the objective values are found. Figure 5 shows the results of our prototype for a proof of concept.

6. CONCLUSION

The presented system for cognitive design computing incorporates methods for integrating various kinds of urban analysis and simulations, based either on stochastic or algorithmic modeling. They are combined in a models container, so that they can be used for the automatic labeling of geometries that are taken either from existing designs or from design synthesis processes. A crucial aspect of the system is the ability to integrate human cognition as a means of enriching and directing the computational design process.
The capabilities of the cognitive design computing system enable an urban planner to treat a planning problem as backcasting problem by defining what a solution should achieve and to automatically query or generate a set of the best possible

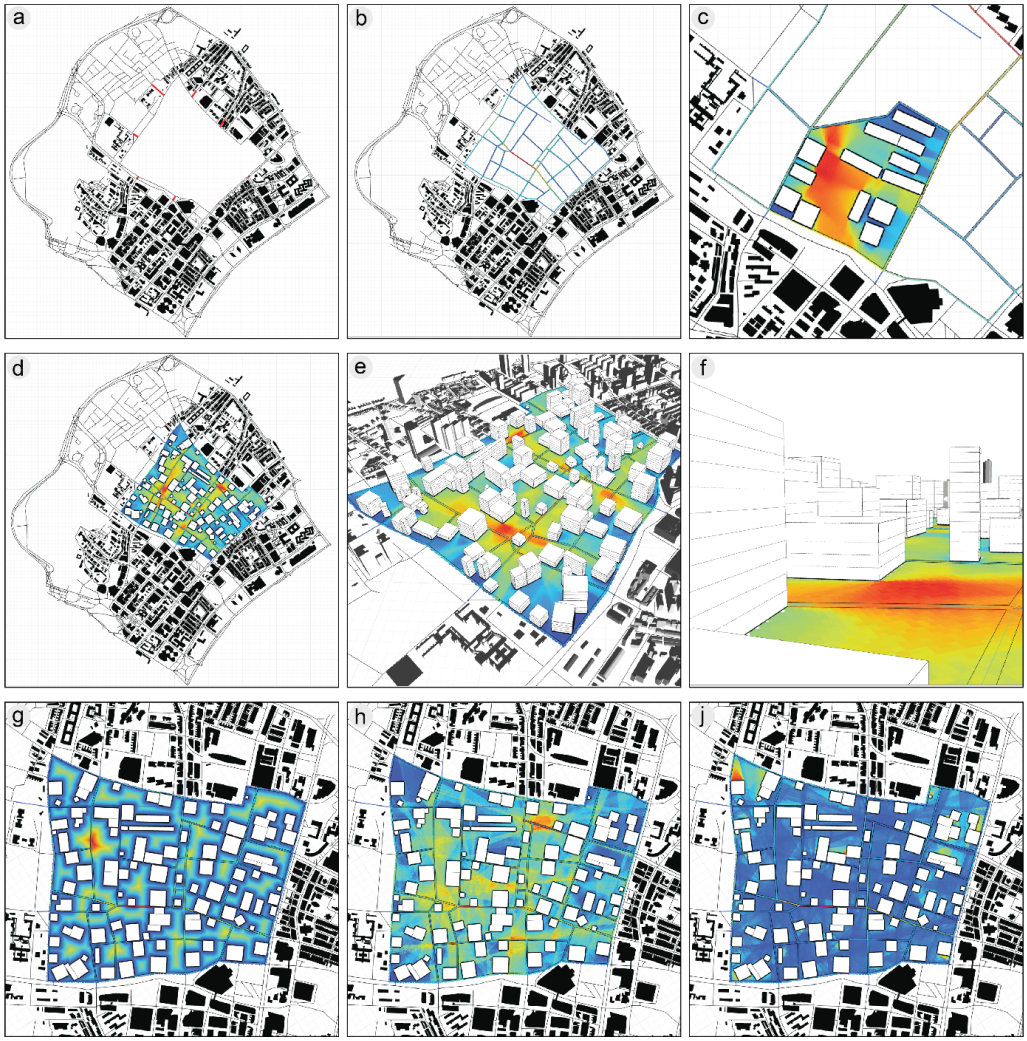


Figure 5: Planning steps: (a) the vacant planning area, (b), the site filled with a generated street network and area Isovist field, (c) a block filled with a generated building layout and area Isovist field, (d) all blocks filled with generated building layouts and area Isovist field, (e) perspective view with area Isovist field, (f) detail of perspective view, (g) min radial Isovist field analysis, (h) occlusivity Isovist field analysis, (j) compactness Isovist field analysis.

solutions. This kind of computational planning process we can call evidence-based planning. It offers proof that the designer meets the original explicitly defined design requirements. This way of thinking offers a new approach for taking command on a computational design process.

Our cognitive design computing system is developed for the specific requirements of the urban planning context, in which planning goals and considered influences change often during the process. In other words, the problem is defined during the planning process. To achieve this, a computational planning support system needs to enable a user to interact with the geometrical elements, change restrictions and objective functions and produce understandable visualizations

during the iterative search process. Because of the close collaboration between computer and designer, we call this approach cognitive design computing. The result is an urban design support system that guides urban planners efficiently through an ever-changing search space, thereby assisting them in finding good compromise solutions for complex planning problems.

For the technical realization of an understandable map of design solutions, we introduced a method based on Self-Organizing-Maps for clustering design variants. The obvious advantage of this arrangement is that it allows us to find similar variants close to each other and helps to clearly identify the number of more distinctly different design solutions since they form separate clusters. The clusters can be also understood as representation of design strategies that can be explored in more detail.

ACKNOWLEDGEMENT

The section on data analysis is based on the PhD of Matthias Standfest and the one on user interaction and learning is based on the PhD of Artem Chirkin. The research presented in this paper was partially conducted at the Future Cities Laboratory at the Singapore-ETH Centre, which was established collaboratively between ETH Zurich and Singapore's National Research Foundation (FI 370074016) under its Campus for Research Excellence and Technological Enterprise program. The research was also funded by the Swiss National Science Foundation (100013L_149552). Thanks to Julian Reisenberger for the language editing.

REFERENCES

Bader, J., & Zitzler, E. (2011). HypE: an algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1), 45–76. http://doi.org/10.1162/EVCO_a_00009

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons. Retrieved from http://books.google.de/books?id=OSTn4GSy2uQC

Derix, C., & Jagannath, P. (2014). Digital intuition – Autonomous classifiers for spatial analysis and empirical design. *The Journal of Space Syntax*, 5(2), 189–215.

Dillenburger, B. (2010). Space Index: A retrieval-system for building-plots. *In 28th Conference on Education in Computer Aided Architectural Design in Europe (eCAADe 2010)* (pp. 893–900). Zurich, Switzerland.

Gibney, E. (2016). Google AI algorithm masters ancient game of Go: Deep-learning software defeats human professional for first time. *Nature*, 529, 445–446. http://doi.org/10.1038/529445a

Gool, P. M. P. W. S. H. A. U. L. Van, Müller, P., Wonka, P., Haegler, S., Ulmer, A., & Van Gool, L. (2006). Procedural Modeling of Buildings. In A. C. M. T. on G. (TOG) (Ed.), *ACM SIGGRAPH* (Vol. 25, pp. 614–623). Boston: ACM Press. Retrieved from http://www.vision.ee.ethz.ch/~pmueller/documents/mueller.procedural_modeling_of_buildings.SG2006.web-version.pdf

Kelly, J. E. I., & Hamm, S. (2013). *Smart Machines: IBM's Watson and the Era of Cognitive Computing*. New York: Press, Columbia University.

Knecht, K., & Koenig, R. (2012). Automatische Grundstücksumlegung mithilfe von Unterteilungsalgorithmen und typenbasierte Generierung von Stadtstrukturen. (D. Donath & R. Koenig, Eds.) *Arbeitspapiere Informatik in der Architektur* (Vol. 15). Weimar: Bauhaus-Universität Weimar. Retrieved from http://infar.architektur.uni-weimar.de/service/drupal-infar/Arbeitspapier15

Koenig, R. (2015a). CPlan: An Open Source Library for Computational Analysis and Synthesis. In B. Martens, G. Wurzer, G. T, W. E. Lorenz, & R. Schaffranek (Eds.), R*eal Time - Proceedings of the 33rd eCAADe Conference* (Vol. 1, pp. 245–250). Vienna: Vienna University of Technology. http://doi.org/10.13140/RG.2.1.2001.9681

Koenig, R. (2015b). Urban Design Synthesis for Building Layouts Urban Design Synthesis for Building Layouts based on Evolutionary Many-Criteria Optimization. *International Journal of Architectural Computing*, 13(3+4), 257–270.

Koenig, R., & Knecht, K. (2014). Comparing two evolutionary algorithm based methods for layout generation: Dense packing versus subdivision. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 28(03), 285–299. http://doi.org/10.1017/S0890060414000237

Koenig, R., & Schneider, S. (2012). Hierarchical structuring of layout problems in an interactive evolutionary layout system. *AIEDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 26(2), 129–142. http://doi.org/10.1017/S0890060412000030

Koenig, R., Treyer, L., & Schmitt, G. (2013). Graphical smalltalk with my optimization system for urban planning tasks. In R. Stouffs & S. Sariyildiz (Eds.), *Proceedings of the 31st eCAADe Conference* – Volume 2 (pp. 195–203). Delft, Netherlands. Retrieved from http://cumincad.scix.net/cgi-bin/works/Show?_id=ecaade2013_197&sort=DEFAULT&search=Koenig Reinhard&hits=10

Langenhan, C., Weber, M., Petzold, F., & Dengel, A. (2011). Investigating research strategies for accessing knowledge stored in semantic models. Proceedings of eCAADe 2011. *Education and Research in Computer Aided Architectural Design in Europe* (eCAADe-2011), 403–410.

Maruani, T., & Amit-Cohen, I. (2007). *Open space planning models: A review of approaches and methods. Landscape and Urban Planning*, 81(1-2), 1–13. http://doi.org/10.1016/j.landurbplan.2007.01.003

Mitchell, J. (1977). *Computer-aided architectural design. Readings*. New York: Van Nostrand Reinhold Company Inc.

Parish, Y. I. H., & Müller, P. (2001). Procedural Modeling of Cities. In *SIGGRAPH* (pp. 301–308). Los Angeles, CA: ACM. http://doi.org/http://portal.acm.org/citation.cfm?id=383292

Radford, A., & Gero, J. S. (1988). *Design by optimization in architecture, building and construction*. New York and Wokingham: Van Nostrand Reinhold.

Rittel, H. W. J., & Webber, M. M. (1973). Dilemmas in a General Theory of Planning. *Policy Sciences*, 4, 155–169.

Rouse, M. (2014). *Cognitive Computing*. Retrieved March 15, 2016, from http://whatis.techtarget.com/definition/cognitive-computing

Schaffranek, R., & Vasku, M. (2013). SPACE SYNTAX FOR GENERATIVE DESIGN : On the application of a new tool. In Y. O. Kim, H. T. Park, & K. W. Seo (Eds.), *Ninth International Space Syntax Symposium* (p. 12). Seoul. Retrieved from http://www.sss9.or.kr/paperpdf/mmd/sss9_2013_ref050_p.pdf

Standfest, M. (2014). Unsupervised Symmetric Polygon Mesh Mapping The Dualism of Mesh Representation and its Implementation for Many Layered Self-Organizing Map Architectures. In E. M. Thompson (Ed.), *32nd eCAADe Conference* (Vol. 1, pp. 505–513). Newcastle upon Tyne, England. Retrieved from http://cumincad.scix.net/cgi-bin/works/Show?_id=ecaade2014_149&sort=DEFAULT&search=/series:"eCAADe"&hits=2298

Stiny, G., & Mitchell, J. (1978). The Palladian Grammar. *Environment and Planning B: Planning and Design*, 5(1), 5–18.

Tonn, C. (2013). *HDRI Sky Generator*. Retrieved May 28, 2015, from http://freac-x.de/freac/drupal/?q=download/hdri_sky_generator

Treyer, L., Klein, B., König, R., & Meixner, C. (2015). Lightweight urban computation interchange (LUCI) system. In *FOSS4G* 2015 Conference (p. 12). Seoul, South Korea: FOSS4G. http://doi.org/http://dx.doi.org/10.3929/ethz-a-010525461

Weber, B., Müller, P., Wonka, P., & Gross, M. (2009). *Interactive geometric simulation of 4D cities* (Vol. 28, pp. 481–492).

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4), 257–271. http://doi.org/10.1109/4235.797969