AUTOMATICALLY GENERATING LAYOUTS OF LARGE-SCALE OFFICE PARK USING POSITION-BASED DYNAMICS

SHUQI CAO[1] and GUOHUA JI[2]
[1,2]*School of Architecture and Urban Planning, Nanjing University, China*
[1]*dg20360001@smail.nju.edu.cn* [2]*jgh@nju.edu.cn*

Abstract. In this paper we propose an automatic layout algorithm using PBD (Position-Based Dynamic) for large-scale office park planning. Typically, the organization of buildings into a layout is a labor-intensive problem, and takes up most of designers' working time. Unlike Evolutionary Algorithms who has high computational cost, and GAN (Generative Adversarial Networks) whose constraints are not explicit, PBD can handle complex geometric constraints fast enough to be used in interactive environments. The high efficiency will not only accelerate the design iteration from draft to drawings, but also provide precious feasible sample for performance optimization. Furthermore, PBD is intuitive and flexible to be implemented which makes it a potential technique to be used in real design workflow.

Keywords. Generative Design; Automated Layout Generation; Position-Based Dynamics; Real-time Design Tool; Exploratory Design.

1. Introduction

Planning an office park, like all architectural layout tasks, is a labor-intensive problem. Besides satisfying complex topological and geometric constraints, designers also need to adjust their layouts constantly to pursuit perfect user experience and optimal environment performance. Obviously, arranging buildings with architectural criteria into a certain site is an iterative and complex task, which becomes ever less impossible as a manual process as the scale increases.

Automated computational techniques are considered to be feasible and necessary to address this dilemma. Multiple algorithms have been successfully developed to automate generation of floorplans, from more conventional ones such as exhaustive search, shape grammar and rule-based methods, to more recent ones such as heuristic search (performance-driven) and machine learning methods (data-driven). However, the existing algorithms, which are either small-in-scale, or time-costing, or loose-constrained, can't work well with the task of organizing a group of buildings on one site, as this task usually has less definite adjacency constraints and more strict geometric constraints (e.g., keeping minimum distance and assuring natural illumination). Additionally, we learned that designers are interested in computer's rapid feedback on whether their concept was feasible or

PROJECTIONS, Proceedings of the 26th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2021, Volume 1, 21-30.

not, and which feasible layouts their concept could be. The requirement places greater demands on the speed of layout algorithms.

This paper proposes a fast and controllable algorithm using Position-Based Dynamic (PBD) (Müller et al. 2007) to layout large-scale office park automatically. This strategy, which formulates layout problem as to position and orient a set of rigid bodies, has several early precedents applied to floorplan layout (Harada et al. 1995). The process of satisfying complex geometric constraints has been accelerated based on laws of dynamics. Then, PBD, one of the most state-of-the-art and universal techniques using computer animation, has further significantly accelerated the process. Meanwhile, the scalability and interactivity of this algorithm also contributes to its application of buildings layout design. Weiss et al. (2018) have applied PBD to indoor scene synthesis, and demonstrated that it can achieve results similar to conventional layout synthesis faster.

We implemented the algorithm in C# as a Grasshopper (GH) component library, and carried out a number of office park layout experiments. The present results show that the method can achieve multiple feasible solutions faster than universal optimization solvers commonly used on GH platform even if the scale is large and the constraints are relatively complex. In brief, the algorithm can be regarded as a potential technique to be implemented and promoted in the real design workflow.

2. Related work

There are years of research and diversity of approaches to various architectural layout problems. Our discussion mainly focuses on design tasks similar to office park planning. They are NP-complete problems, and early studies have shown that exhaustively enumerating cannot handle the exponential growth of possible arrangements as the size of the problem increases (Kalay 2004, p.241). Some procedural methods (Yang 2013) can model visually plausible urban scene, but they scarcely could satisfy architectrual criteria. Researchers have realized that the main challenge of this problem stems from the great number of geometric variables and complex constraints, which can only be solved using AI technology such as intelligent optimization algorithm and machine learning.

Genetic algorithms (GA), Evolutionary Strategy(ES), and other Evolutionary Algorithms (EAs), are regarded as a powerful tool for design explorations of performance-driven geometry in architectural design (Turrin 2011). As we know, most early applications such as Kämpf et al. (2010) have been limited to highly codified and regular basic templates. Yi and Kim (2015) successfully used GA to optimize the solar right of tall apartments with complex shape at the expense of tackling only 5 buildings. Compare with single-objective, multi-objective optimization has been valued more practical worth for synthetical consideration in design (Kim and Yi 2019). However, countless Pareto optimals just lead more inoperability to this task, and researchers in this field have to figure out how to interact with their user to select the “optimal” (Vierlinge 2018).

EAs can also be used to solve the numerous geometric constraints, and have proved their innovative abilities in part through their application to the generation

of floor plans (Rodrigues et al. 2017). Simulated Annealing (Merrell et al. 2010), Markov Chain Monte Carlo (Yeh et al. 2012) and Multi-Agent System (MAS) (Guo and Li 2017) are widely-used methods too. Nevertheless, all of these non-gradient stochastic optimization algorithms are extremely tedious to be implemented and expensive to be calculated. Therefore, few architects apply these methods to indeed support their design and creation process.

Making computational design process comes alive, interactive and exploratory might play a more utilitarian role for designers' creative work. As early as 1999, Arvin and House (1999) had introduced the concept of responsive design, and applying physically based modeling techniques to space layout planning. MAS is considered to be potential for interactivity with the same reason. But when it comes to real time response and knowing what designers want to do, existing instances are not persuasive enough up to now.

With the tide of Deep Learning in recent years, Data-driven generative design has back into view. The cases provided by Autodesk (Davis 2019), Spacemaker.ai (Chaillou 2019), and XKool Group has shown the surprising abilities of GAN (Generative Adversarial Networks). Although they can instantaneously convert a very simple draft to a complete drawing, the inherent weakness of these approaches cannot be denied–they are precedent dependent, i.e., we can't assure the accuracy of GAN when we just change a simple constraint different from its examples.

But the power of Machine learning is still inspiring. Accordingly, we believe that some state-of-the-art technologies in other fields can reinvigorate aged method. PBD is one of the most popular methods of nowadays interactive computer graphics community, and its superiority in layout synthesis has been proved. We propose the technique can play a more pragmatic role in computational design process.

3. Position-Based Layout

3.1. PROBLEM DEFINITIONS

An office park layout problem can be modeled as to position and orient a given number of buildings with different dimensions onto a given site while strictly satisfies the code, and then to find the arrangement which satisfies functional and aesthetic criteria given by the designer as much as possible. Thus, we express the buildings we need to arrange as a set of n buildings $B = \{b_1, b_2, \ldots, b_n\}$. Each $b_i \in B(i = 1, 2, \ldots, n)$ has (1) a position p_i, (2) an orientation θ_i, (3) an associated building shape mesh, (4) a set of associated agent model, and (5) an inverse mass w_i. The w_i is the reciprocal of the building's mass m_i, and the m_i is determined by the volume of the building shape mesh. $w_i = 0$ means that the building is immovable with infinite mass.

We express building code, and other design criteria as a set of m constraints $D = \{d_1, d_2, \ldots, d_m\}$. Each $d_j \in D(j = 1, 2, \ldots, m)$ consists of

- the set of nj indices $\{i_1, i_2, \ldots, i_nj\}, i_k \in [1, 2, \ldots, n]$ to find the buildings the constraint works with,
- a stiffness $k_j, 0 \leq k_j \leq 1$, and

- a function $C_j\left(p_{i1}, p_{i2}, \ldots, p_{i(nj)}\right)$ with the type of either equality or inequality.

If we measure the quality of a layout by how much the constraints are satisfied, we can introduce the conception of elastic potential energy. By loosely applying Hook's law, the potential energy may be considered as proportional to the square of embed-ding depths, so we employ the energy function $E = \sum_{j=1}^{m} \gamma_j C_j^2$,where γ_i is the respective weight of the constraints.

Obviously, it's a non-convex optimization problem, while the compulsory constraints make it thornier. From the review in previous section, we learn that designers hate EAs' inefficient and heavy-to-implement. For real-time interaction, we seek help from the most popular technique in interactive computer graphic, and is inspired by the lowest energy principle of the rigid bodies dynamics system (the potential energy will decline to the local lowest as the system convergences to its steady state). We find that the process of organizing buildings inside a boundary to satisfy multiple constraints is similar to the process of putting rigid bodies in a container to equilibrate various force. Hence, we propose that animation physics tech might help.

We first realized a simple prototype using Kangaroo, a live physics engine on GH, before we turned to PBD. For the same case (see Site1 in Section 4) , Kangaroo produced satisfactory results faster than Galapagos (based on ES and SA) (see top in figure 1). However, when we tested the more complex cases, we observed obvious drawback of Kangaroo in solving layout problem. First, in the Kangaroo's simulation process, some potential energy converts to velocity, which is redundancy. Second, owing to the collision constraints, one object can hardly across another to reach its more ideal position, which means the local optima. Third, it's difficult to represent some design criteria by directly using Kangaroo's existing constraints.

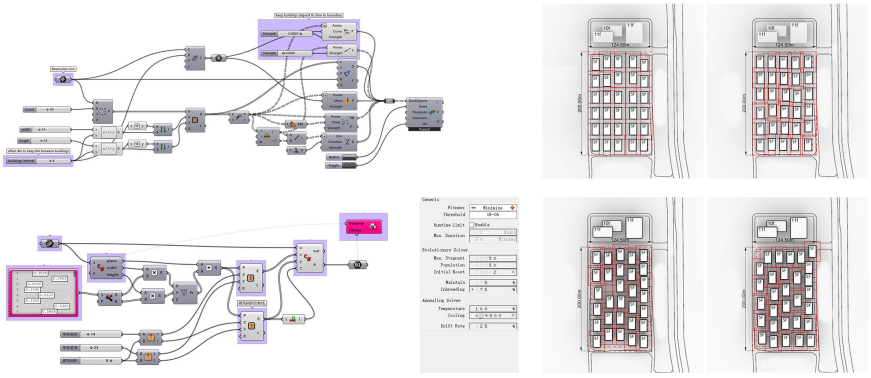

Figure 1. Top:settings and results using Kangaroo. Bottom:settings and results using Galapagos.

PBD, which is proposed by Muller, contributes to overcoming these faults. PBD omits the velocity layer from the traditional acceleration-velocity-displacement time integration, and immediately correct the position based on the solution of a quasi-static problem. Despite not as accurate as force-based methods, position-based approaches are fast, stable and controllable, which make them well-suited for use in interactive environment (Bender 2016). As we only care for the system's steady state, we can further ignore all the velocities, and directly work on the positions.

3.2. ALGORITHM OVERVIEW

In this Section we will overview the algorithm , and only explain the issues what is important or different from original PBD. Our algorithm is as follows:

```
(1) forall Buildings b_i do Randomly Initialize p_i, theta_i
(2) for SolverIteration=0 to Max do:
(3)    UpdateStiffnesses(C_1,...,C_m)
(4)    UpdateAgentModel(AM_1,...,AM_n)
(5)    GenerateCollisionConstraints(b_1,...,b_n)
(6)    if no collision constraints generated then return
(7)    ProjectConstraints(C_1,...,C_m)
```

Lines (1) just randomly initialize the position and orientation of buildings, for generating various and novel solutions. **Lines (2)-(7)** is an iterative solver which tried to adjust the positions of buildings or other points associated to the buildings to satisfy all the constraints. PBD applies the idea of Gauss-Seidel-type iteration (GS) to solve each constraints using the Project Constraints Function (**Line(7)**) independently one after the other, i.e., positions change immediately get visible to the process. Muller demonstrated the mechanism why GS can significantly speed up convergence in his paper, and suggested to solve the constraints in a random order to avoid oscillations.

In **Line(7)**, the Project Constraints Function finds the correction $\Delta\mathbf{p}$ to make $C(\mathbf{p}+\Delta\mathbf{p}) = \mathbf{0}$ in the fastest speed. The $\mathbf{p}$ is a set of positions of not only the buildings but the points associated to the buildings. Because only positions will change the value of constraints $C(\mathbf{p})$, we directly correct p instead of the orientation in the function. The gradient $\nabla_{\mathbf{p}}C(\mathbf{p})$ is the direction of maximal change, A first-order Taylor approximation $C(\mathbf{p}+\Delta\mathbf{p}) \approx C(\mathbf{p})+\nabla_{\mathbf{p}}C(\mathbf{p})\cdot\Delta\mathbf{p} = \mathbf{0}$ is used to linearize the constraint. The final displacement vector is determined by $\Delta\mathbf{p}_i = -s\nabla_{\mathbf{p}}C(\mathbf{p})$, where $s = \dfrac{w_i C(\mathbf{p})}{\sum_{i=1}^{n} w_i ||\nabla_{\mathbf{p}}C(\mathbf{p})||^2}$. Here, we also need a stiffness k obtained in **line (3)** to control the priority of different constraints. The process is analogous to finding local optimal solution using gradient descent, but the random initialization has provided a global view. In the following modifying, either with user or not, the drastic change of layout is not expected.

In **Line (3)**, we accepted Weiss's proposal to use dynamic stiffness. Starting from a random initial layout, stiffness of the collision constraints increases over time, and others decrease.

Line (4) updates the agent model of each building according to its real time position. In real planning task, besides the building entities, some virtual volumes, which be called agent models, also should be no overlap and fit other rules. A building might have several corresponding agent models which will change as the building moves, such as volume agent model, distance agent model, view field agent model, shadow agent model, and so on (see in figure 2). At present, all of our constraints are collision constraints between different agent models.

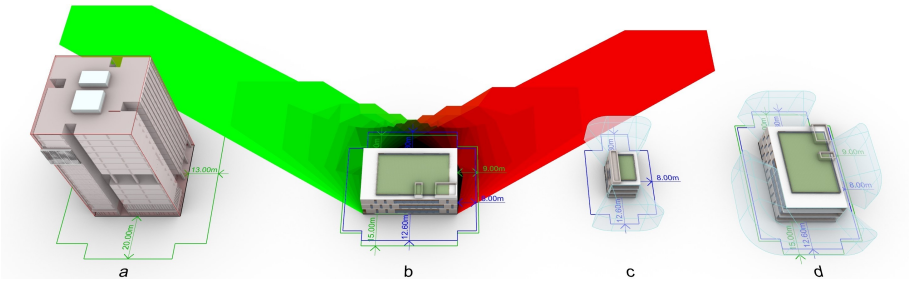

Figure 2. The greean and bule cruves are the outlines of distacne agent models which will be updated in different stations, and each building is showing its a:volume agent model. b: shadow agent model with time information represented by colour. c,d:view field agent model.

Line (5) generates collision constraints. Solving collision constraints is not different from solving general constraints, but there are many sophisticated collision detection approaches in the field of computer graphic. Generating collision constraints independently can accelerate the computation and reduce the number of constraints in **Line (7)**.

Line (6) checks whether all the compulsory constraints are satisfied, and terminates the iteration if true. We cannot assure a solution to an over-constrained situation, but our tests yet, even whose floor area rates and building densities are slightly over the limitation, can all be successfully solved in an acceptable time.

4. Case Studies

We selected two sites to test the ability of PBD method to deal with the layout tasks with regular boundary, small scale but compact constraints, and the layout tasks with irregular, large-scale and relatively compact constraints.

Site 1 is a completed office park in Dongguan City, Guangdong Province, China. Its boundary is almost a perfect rectangle, 124.5m wide, 200m long, with a total area of $2.49hm^2$. Its east side is adjacent to a road, and the location of its entrances are determined (see in figure 3a-3c). The local code limits the buildings setback line and the intervals between various types of products. We take the existing design as the baseline (see in figure 3d, 3e), and 3 types of products should be arranged. **15 Office Houses** with a floor area between $240 - 500m^2$, 4 floors, a height of 19m and a building area of no more than $1,600m^2$; **2 Semi-detached Office Houses** with a plane size of $30m \cdot 20m$, 4 floors and a height of 19m; **3 Multi-layer Office Buildings** with a plane size of $21m \cdot 52m$, 6 floors and a height

of 28m. Then, the plan has a total floor area of $9,476m^2$, a total buildings area of $44,456m^2$, a building density(BD) is 38%, and a Floor Area Ratio (FAR) of 1.78.

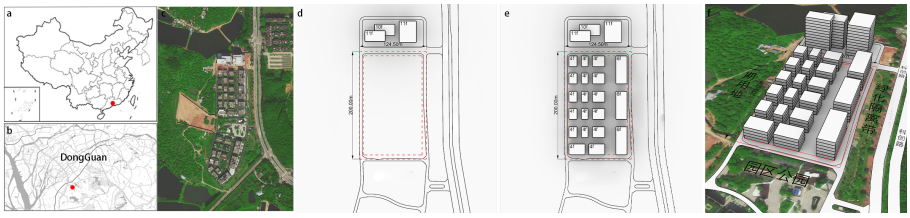

Figure 3. Information of Site 1.

Each product in our experiment is slightly smaller than the baseline, with BD of 35.9% and FAR 1.68. Using the smaller dimensions is a strategy for producing diverse results, and the results can easily exceed the baseline by further detail optimization. In a set of 800 tests, the average time per test was 11.18s, of which 798 were feasible results. Some of the results are shown in figure 4.

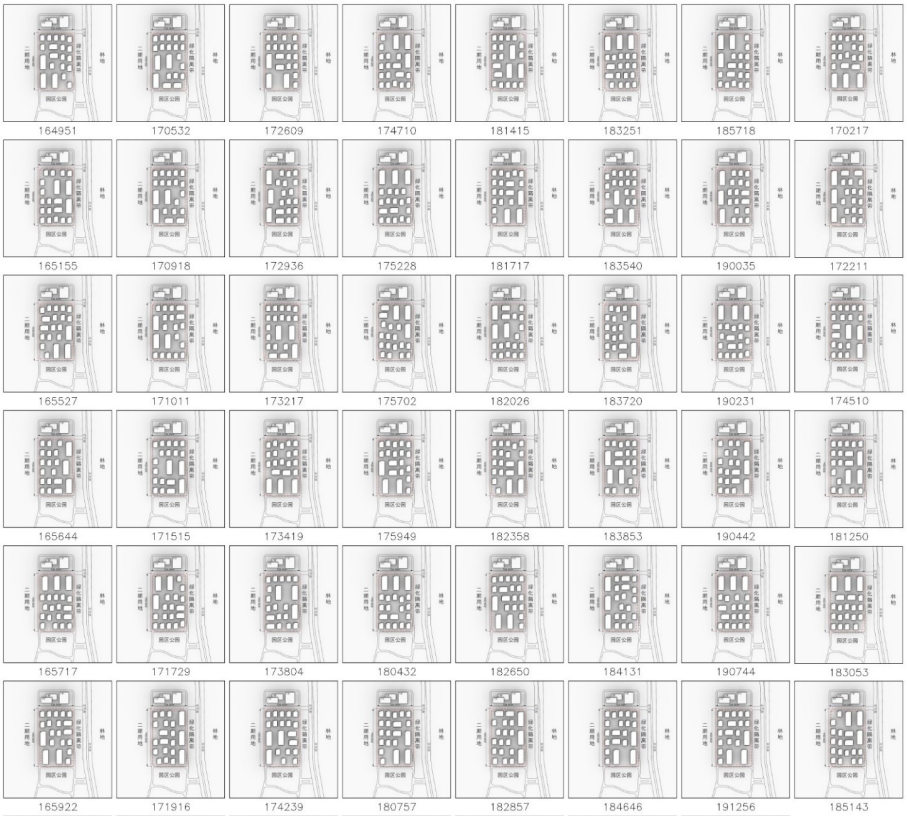

Figure 4. Some results of Case 1.

Site 2 is an undeveloped site in Danyang City, Jiangsu Province, China. Its boundary is very irregular, with an area of $12.6hm^2$. Its south side and west side are adjacent to the road, a residential area is on its north side, and its completed Phase I project is on its west side across a landscape lake. In addition to more complex boundary and interval limitation, this site requires a ceremonial entrance landscape in the southwest corner, two motor vehicle entrances settled autonomously, and an arrangement coordinated and connected with the phase I project. We manually draw control lines for these requirements (see in figure 5).

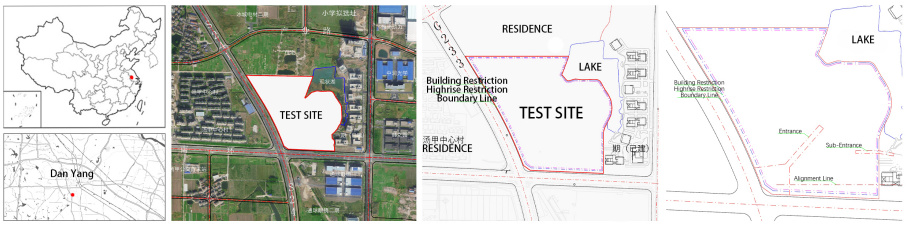

Figure 5. Information of Site 2.

A total of 92 products of 8 different types need to be arranged in the plot with the total building area of $107572m^2$, the total floor area of $28193m^2$, the FAR is 0.86, and the BD of 22.6% (see in figure 6).

PRODUCT TYPES	Office House1	Office House2	Semi-detached Office1	Semi-detached Office2	Semi-detached Office1	Multi - Layer Office	Highrise Office	Highrise Apartment
Building Massing								
Dimensions	11m*12m	18m*10m	24m*12m	30m*12m	30m*14m	40m*24m	35m*35m	32m*32m
Height	12.6m	12.6m	12.6m	12.6m	12.6m	20.7m	51.3	38.7m
Building Area	132 m²	180 m²	288 m²	360 m²	420 m²	960 m²	1225 m²	1024 m²
Floor Area	396 m²	540 m²	864 m²	1080 m²	1260 m²	3840 m²	14700 m²	10240 m²

Figure 6. Types, dimensions and quantities of the buildings should be arranged in Case 2.

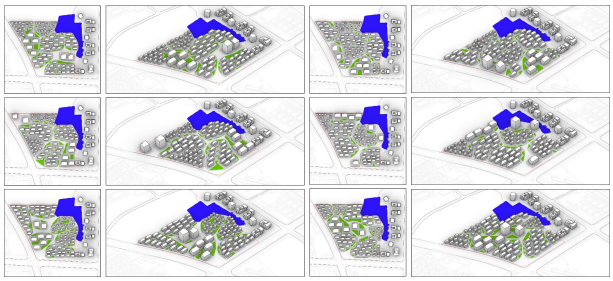

Figure 7. Some results of Case 2.

We added additional constraints to regularize the cluster boundary, and finally screened out the part of the automatically adjusted results (see in Figure 7), which

showing an overall diversity. Although the results still cannot be directly used as design, through the analysis and comparison of these possibilities provided by the computer, designers can draw many valuable preliminary design judgments.

5. Discussion

Our experiments have preliminary shown the advantage of Position-Based Methods using in Large-scale Layout. At present, it can organize a big number of buildings on a continuous planar space at low computational cost. Architectural design is a repetitive and iterative work. The high efficiency of converting from draft to drawings will accelerate the design iteration, and then contribute to the whole design process. At the same time, from random initial states, PBD can generate abundant solutions for comparison to achieve the layout with higher quality.

In our tests on the GH platform, PBD is faster than Galapagos by at least an order of magnitude with more satisfactory solutions, for the reason that it uses the gradient information what stochastic optimization solvers do not use. Compared with the faddish methods based on GAN, PBD provides more exact, multiple and scalable constraints, which makes the solutions provide more valuable information for design.

Another significant superiority of PBD is its flexible interaction mode. Working in akin way as Kangaroo (see in figure 8), our components can be smoothly understood and implemented by the designers who are familiar to GH, and the developer can easily promote or expand some functions of components separately. Furthermore, it is intuitive and flexible to define the buildings or constraints using our components. The conception should be simple from the user's point of view that moving buildings based on design intentions just like moving physical objects. The mechanism of PBD also supports to add, subtract or modify both buildings and constraints during the generation process. All of these make PBD more suitable in practical design environment.

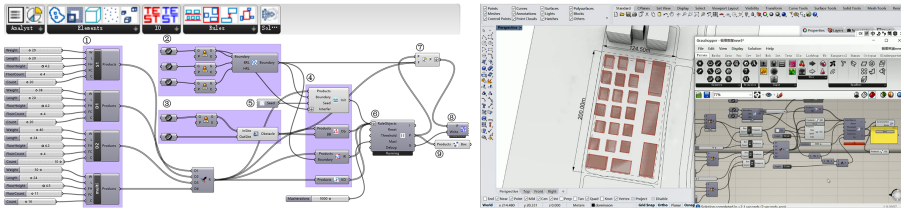

Figure 8. User Interface.

However, we cannot deny the inherent limitations of PBD that not all the design conceptions can be easily converted to position-related constraints. In the present study, we encoded some basic constraints relevant to layout design, and we will deal with layout constraints as broader as possible in our future work. Moreover, PBD could not optimize environmental performance, but it could help the search of precious feasible sample for stochastic optimization.

In our future work, more types of elements such as terrain, roads and

comprehensive buildings also should be introduced to our system. To optimize the user interface of inputting and manipulating design elements is also an interesting issue. From technical perspective, incorporating GPU parallelization or applying hierarchical approaches can further speed up the procedure, and how to update the stiffness in solution iteration is still an immature and ambiguous problem need to be explored.

References

Arvin, S.A. and House, D.H.: 1999, Making Designs Come Alive: Using Physically Based Modeling Techniques in Space Layout Planning, *Computers in Building: Proceedings of the CAADfutures'99*, Georgia Institute of Technology, Atlanta, Georgia, USA, 245-262.

Bender, J.: 2016, "PositionBasedDynamics" . Available from <https://github.com/InteractiveComputerGraphics/PositionBasedDynamics> (accessed 2rd December 2020).

Chaillou, S.: 2019, "ArchiGAN: a Generative Stack for Apartment Building Design" . Available from <https://developer.nvidia.com/blog/?p=15310> (accessed 2rd December 2020).

Davis, M.: 2019, "Could Machine Learning Improve Work-Life Balance in the Future of Architecture?" . Available from <https://redshift.autodesk.com/future-of-architecture/> (accessed 2rd December 2020).

Guo, Z. and Li, B.: 2017, Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system, *Frontiers of Architectural Research*, **6**(1), 53-62.

Harada, M., Witkin, A. and Baraff, D.: 1995, Interactive Physically-Based Manipulation of Discrete/Continuous Models, *SIGGRAPH '95*, New York, NY, United States.

Kalay, Y.E.: 2004, *Architecture*, MIT Press, Cambridge, MA, the United States.

Kim, H. and Yi, Y.K.: 2019, QuVue implementation for decisions related to high-rise residential building layouts, *Building and Environment*, **148**, 116-127.

Kämpf, J.H., Montavon, M., Bunyesc, J., Bolliger, R. and Robinson, D.: 2010, Optimisation of buildings' solar irradiation availability, *Solar Energy*, **84**(4), 596-603.

Merrell, P., Schkufza, E. and Koltun, V.: 2010, Computer-generated residential building layouts, *ACM SIGGRAPH Asia 2010*, Seoul, South Korea, Association for Computing Machinery: Article 181..

Müller, M., Heidelberger, B., Hennix, M. and Ratcliff, J.: 2007, Position based dynamics, *Journal of Visual Communication and Image Representation*, **18**(2), 109-118.

Rodrigues, E., Sousa-Rodrigues, D., Teixeira de Sampayo, M., Gaspar, A.R., Gomes, A. and Henggeler Antunes, C.: 2017, Clustering of architectural floor plans: A comparison of shape representations, *Automation in Construction*, **80**, 48-65.

Turrin, M., Buelow, P.v. and Stouffs, R.: 2011, Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms, *Adv. Eng. Inform.*, **25**(4), 656-675.

Vierlinge, R.: 2018, "Octopus | Food4Rhino" . Available from <https://www.food4rhino.com/app/octopus> (accessed 2rd December 2020).

Weiss, T., Litteneker, A., Duncan, N., Nakada, M., Jiang, C., Yu, L. and Terzopoulos, D.: 2019, Fast and Scalable Position-Based Layout Synthesis, *IEEE Transactions on Visualization and Computer Graphics*, **25**(12), 3231-3243.

Yang, Y.L., Wang, J., Vouga, E. and Wonka, P.: 2013, Urban pattern: layout design by hierarchical domain splitting, *ACM Trans. Graph.*, **32**(6), Article 181.

Yeh, Y.T., Yang, L., Watson, M., Goodman, N.D. and Hanrahan, P.: 2012, Synthesizing open worlds with constraints using locally annealed reversible jump MCMC, *ACM Trans. Graph.*, **31**(4), Article 56.

Yi, Y.K. and Kim, H.: 2015, Agent-based geometry optimization with Genetic Algorithm (GA) for tall apartment's solar right, *Solar Energy*, **113**, 236-250.