Fast, Scalable Layout Synthesis

Tomer Weiss, Alan Litteneker, Noah Duncan[*], Chenfanfu Jiang[†], Lap-Fai Yu[‡], Demetri Terzopoulos

Abstract

The arrangement of objects in an interior layout can be challenging for non-experts, as is affirmed by the existence of design professionals. Most research into the automation of this task has yielded methods that can synthesize layouts of objects respecting aesthetic and functional constraints that are non-linear and competing. These methods usually adopt a purely stochastic scheme, which samples from different layout configurations, a process that is slow and inefficient. We propose an alternative physics-based, continuous layout synthesis technique, which results in a significant gain in speed and is readily scalable.

1. Introduction

In this abstract, which summarizes [6], we focus on interior layout synthesis, in which a set of objects is to be arranged in an open space. For example, to find a desirable layout for a living-room, one must consider the visibility of the television, a suitable separation of sofas, and access to adjacent rooms, among other factors that differ according to taste and style. The goal is to position and orient the objects such that they satisfy several functional and aesthetic criteria.

Recently, researchers have proposed several methods for synthesizing layouts that pose layout synthesis as a highly non-convex optimization problem subject to numerous constraints [8, 4, 7]. Due to the complex nature of these problems, previous work applied a probabilistic scheme to sample viable layout candidates. Stochastic methods, such as Markov chain Monte Carlo (McMC) methods [1] are preferred because the constraints are often difficult to express as differentiable functions. Unfortunately, the majority of these techniques become inefficient when confronted with large numbers of objects.

To overcome this problem, we propose a continuous framework for layout synthesis based on Position-Based Dynamics (PBD) [5], a framework designed to simulate physical models in real time scenarios. Our main observation is that there are commonalities between the elastic simulation of deformable objects and layout synthesis. Elasticity penalizes the deformation of an object—the energy increases proportionally to the magnitude of the deformation—and layout synthesis penalizes the magnitude of constraint violation. Both can be expressed as optimization problems, and both can be tackled using continuous optimization procedures. Our new, continuous approach enables the fast generation of dense large-scale layouts that are intractable using previous approaches.

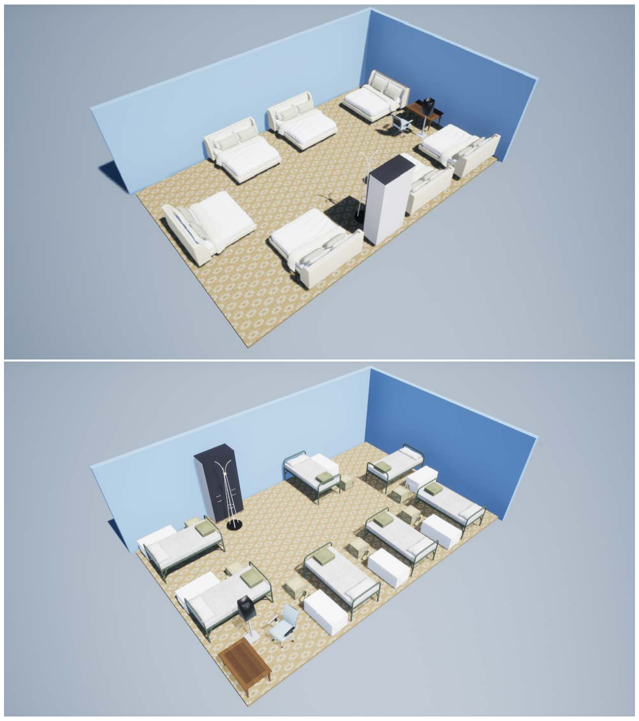
Figure 1: A tightly-packed bedroom. Our method produces different layout suggestions by starting from different random initial conditions.

2. Method

Our method takes as input an environment, a set of objects, and a set of layout constraints. After randomly initializing the orientations and positions of the objects, our method then iteratively modifies the state of each object so

T. Weiss, A. Litteneker, D. Terzopoulos are with the University of California, Los Angeles.
[*] N. Duncan is with Clovi, Inc.
[†] C. Jiang is with the University of Pennsylvania.
[‡] L.-F. Yu is with the University of Massachusetts Boston.

as to satisfy competing constraints, reaching convergence when all constraints are adequately satisfied.

2.1. Setup

A layout is represented by a set of N oriented particles and M constraints. Each particle i, which determines the center of a corresponding 3D object mesh, has a position p_i, an orientation θ_i, a mass m_i, and corresponding inverse mass $w_i = \frac{1}{m_i}$.

A constraint consists of the number n of participating particles, a scalar constraint function C, a stiffness parameter k (where $0 \leq k \leq 1$), and a constraint type (either *unilateral* or *bilateral*).

A bilateral constraint is defined as

$$C(p_1, ..., p_n, \theta_1, ..., \theta_n) = 0,$$

while a unilateral constraint is defined as

$$C(p_1, ..., p_n, \theta_1, ..., \theta_n) \geq 0,$$

where the arguments of both functions denote the positions and orientations of the particles involved in the constraint.

2.2. Layout Synthesis

Given a constraint C and a particle i which is involved in the constraint, we derive a positional correction Δp_i applied to particle i. According to Muller et al. [5], this correction is derived by approximating each constraint function using

$$C(p_i + \Delta p_i) \approx C(p_i) + \nabla C(p_i) \cdot \Delta p_i. \quad (1)$$

We employ PBD's mechanism for solving a layout's positional constraints via approximation, with two different approaches for satisfying constraints. Each constraint is either solved independently and projected, or is solved in a batch with other constraints, averaged together, and then projected. In the batched case, we average the positional correction by the number of constraints affecting the layout item, with a delta averaging coefficient of 1.2, as suggested by Macklin et al. [3].

For constraint C, the positional correction for particle i is

$$\Delta p_i = -s k w_i \nabla_{p_i} C(p_1, ..., p_n), \quad (2)$$

with scaling factor

$$s = \frac{C(p_1, ..., p_n)}{\sum_i w_i \left\| \nabla_{p_i} C(p_1, .., p_n) \right\|^2}, \quad (3)$$

and stiffness parameter k that is adjusted at each iteration. The stiffness of each constraint determines the importance of that constraint; i.e., lower stiffness leads to smaller positional correction. We modified the original stiffness formula suggested by Muller et al. [5] to $k' = 1 - (1-k)^{M/n_s}$, where n_s is the iteration number and $M \geq 1$. Parameter M determines the rate by which the constraint C approaches 0.

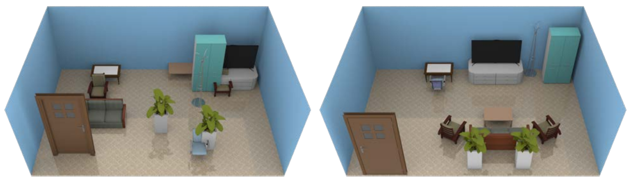
Figure 2: Our method produces different layout suggestions (right) by initializing random initial conditions (left).

2.3. Constraint Types

In order to fit the problem of layout synthesis into a PBD framework, we have adapted a number of common aesthetic layout standards [2] into PBD compatible constraints, only a subset of which are briefly described here. Please refer to [6] for a complete set of layout constraints and more thorough descriptions.

2.3.1 Pairwise Distance

In interior design, two furniture items i and j (e.g., a chair and a table) are often required to be at a certain distance from each other in order for the layout to be deemed comfortable. We utilize the bilateral stretching constraint from PBD to represent a desired distance d between particles i and j as

$$C(p_i, p_j) = \|p_i - p_j\| - d. \quad (4)$$

2.3.2 Heat Point

For a group of objects, the *heat point* is the desired position of the objects' center of mass (eg., recliners around a television). Given a user supplied heat point position $\hat{p}$, this is represented as a bilateral constraint where

$$C(p_1, p_2, ..., p_n) = \left\| \frac{\sum_{j=1}^{n} p_j m_j}{\sum_{j=1}^{n} m_j} - \hat{p} \right\|^2. \quad (5)$$

2.3.3 Collision

As most users want realistic layouts, we generally want to avoid overlapping or colliding objects. To that end, we check for collisions between objects and, if found, resolve each collision as a unilateral distance constraint between the bounding spheres that best fit the objects' underlying meshes.

Since checking for all pairwise object collisions is computationally expensive, we employ a spatial hash in order to reduce the number of collision checks.

2.3.4 Traffic Lanes

Objects should be arranged in a layout that accommodates traffic lanes, space between objects to allow easy access

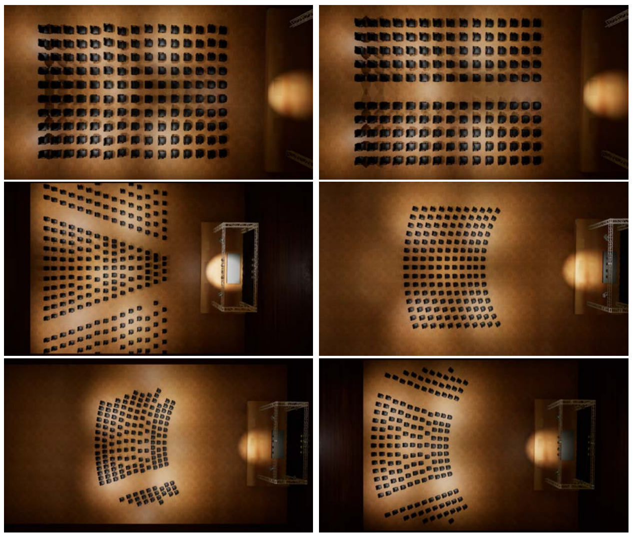
Figure 3: A variety of theater layouts synthesized with our method.

(e.g., different seating tiers) [2]. To this end, we define a clearance between a vector projected from an object at an angle towards another object. This clearance is implemented as a unilateral distance constraint between p_j and a vector v, which starts at p_i and points in direction θ, as

$$C(p_i, p_j) = \left\| p_{v_{\text{proj}}} - p_j \right\| - d, \tag{6}$$

where $p_{v_{\text{proj}}}$ is the point on vector v that is closest to p_j, and $d > 0$ is the desired distance from vector v. Note that $p_{v_{\text{proj}}}$ depends both on the p_i and p_j, and is recalculated if p_i or p_j change. We consider $p_{v_{\text{proj}}}$ a *ghost* particle that is rigidly attached to p_i. Hence, any positional correction that impacts $p_{v_{\text{proj}}}$ is applied to p_i. This constraint creates the effect of pathways; e.g., in the theater example of Fig. 3.

2.3.5 Parenting and Grouping

Layout items can belong to different groups, such as seating tiers in a theater, or chairs around a table. In our framework, a set of objects may be represented by a group particle. Multiple group particles may have constraints expressed between them, or be grouped into a super group particle.

Furthermore, we can hierarchically define internal group layout constraints. For example, suppose a user desires a set objects to be arranged along a curve (eg., the seating tiers in Fig. 3). As each group member is associated with a point on the curve, we can use the distance along the curve to derive an ordering of objects that can accelerate the solving of other internal group constraints, such as pairwise distance.

3. Results

Our method is implemented in Python and Cython on a 2.5 GHz Intel i7 Macintosh system. For all our demos, the initial object locations and orientations were set randomly (Fig. 2). The method terminates the iterative procedure when there has been no improvement upon the best observed layout energy for 50 iterations.

We demonstrated our method's efficacy with experiments that include a tightly-packed bedroom and a theater scene with various seating arrangements. Additionally, we also compared our run-time with that of a baseline McMC method, based on code obtained from the authors of [8]. The tightly-packed bedroom contained 12 objects (Fig. 1), and ran 0.67 seconds versus 22.31 seconds for the baseline McMC approach. The theater (Fig. 3) contained between 169 to 246 objects, and ran 0.48 to 39.5 seconds versus at least 5852 seconds for the baseline McMC approach that managed to terminate.

4. Conclusion

We proposed an alternative, fast and scalable method for layout synthesis, inspired by Position-Based Dynamics (PBD). Our novel method enables interactive layout synthesis, which was slow to intractable using previous stochastic methods, especially on dense layouts with many objects.

References

[1] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, 1995. 1

[2] C. Deasy and T. E. Lasswell. *Designing Places for People*. Whitney, 1990. 2, 3

[3] M. Macklin, M. Müller, N. Chentanez, and T. Kim. Unified particle physics for real-time applications. *ACM Trans Graph*, 33(4):104, 2014. 2

[4] P. Merrell, E. Schkufza, Z. Li, M. Agrawala, and V. Koltun. Interactive furniture layout using interior design guidelines. In *ACM Trans. Graphics*, volume 30, page 87. ACM, 2011. 1

[5] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *Virtual Reality Interactions and Physical Simulations (VRIPHYS)*, 18(2):109–118, 2007. 1, 2

[6] T. Weiss, A. Litteneker, N. Duncan, C. Jiang, L.-F. Yu, and D. Terzopoulos. Fast, scalable layout synthesis. *IEEE Trans. Vis. Comp. Graph.*, 2018. Under review. 1, 2

[7] Y.-T. Yeh, L. Yang, M. Watson, N. D. Goodman, and P. Hanrahan. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Trans. Graphics*, 31(4):56, 2012. 1

[8] L.-F. Yu, S. K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. Osher. Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graphics*, 30(4):86, 2011. 1, 3