UNIVERSITY OF CALIFORNIA

Los Angeles

A Constraint-Based Approach to Crowd Simulation and Layout Synthesis

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Tomer Weiss

2018

ABSTRACT OF THE DISSERTATION

A Constraint-Based Approach
to Crowd Simulation and Layout Synthesis

by

Tomer Weiss
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2018
Professor Demetri Terzopoulos, Chair

Position-based methods have become popular for real-time simulation in computer graphics. In contrast to traditional simulation methods, which are based on Newtonian dynamics, particularly forces, a Position-Based Dynamics (PBD) method computes the positional changes directly, based on a set of well-defined geometric constraints. Therefore, position-based methods are reputed to be more controllable, stable, and faster, which make them well-suited for use in interactive environments. This thesis introduces position-based approaches to addressing the important tasks of virtual crowd simulation and virtual layout synthesis.

For crowd simulation, we introduce a novel method that runs at interactive rates for up to hundreds of thousands of agents. Our method enables the detailed modeling of per-agent behavior in a Lagrangian formulation. We model short-range and long-range collision avoidance to simulate both sparse and dense crowds. On the particles representing agents, we formulate a set of positional constraints that can be readily integrated into a standard PBD solver. We augment the tentative particle motions with planning velocities to determine the preferred velocities of agents, and project the positions onto the constraint manifold to eliminate colliding configurations. The local short-range interaction is represented with collision and frictional contact between agents, as in the discrete simulation of granular materials. We incorporate a cohesion model for simulating collective behaviors and propose a new constraint for dealing with potential future collisions. Our method is suitable for use

in interactive games.

For layout synthesis, we propose a position-based interior layout synthesis method that is able to rapidly synthesize large scale layouts that were previously intractable. An interior layout modeling task can be challenging for non-experts, hence the existence of interior design professionals. Recent 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 a distribution of layout configurations, a process that is slow and inefficient. We introduce an alternative physics-based, continuous layout synthesis technique, which results in a significant gain in speed and is readily scalable. We demonstrate our method on a diverse set of examples and show that it achieves results similar to conventional layout synthesis based on a Markov chain Monte Carlo (McMC) state-search step, but is faster by at least an order of magnitude and can handle layouts of unprecedented size and tight layouts that can overwhelm McMC.

The dissertation of Tomer Weiss is approved.

Song-Chun Zhu

Joseph M. Teran

Stanly J. Osher

Demetri Terzopoulos, Committee Chair

University of California, Los Angeles

2018

To my family, friends, ...

and everyone else who supported me.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

First, I would like to thank my advisor, Professor Demetri Terzopoulos. I am truly fortunate that I got to know him. Even though he was not my initial PhD advisor, he gracefully accepted me into his lab during the second year of my PhD program. He provided unconditional support and advice whether I needed it, and for that I am grateful. His advice was always given willingly and cheerfully. Without his leadership and guidance, I would not have accomplished the results reported in this thesis. He is my role model as a researcher and a research leader.

Secondly, I would like to express my appreciation to the other members of my PhD thesis committee, Professor Stanley Osher, Professor Joseph Teran, and Professor Song-Chun Zhu. The various insights from their course lectures and their feedback during my oral qualifying exam and other interactions was valuable in advancing my research.

Additionally, I would like to thank all my fellow members of the UCLA Computer Graphics and Vision Laboratory and other colleagues. Their companionship made my PhD experience extraordinarily fulfilling, and my time in Los Angeles and at UCLA very pleasant. In particular, I thank my current lab colleagues Masaki Nakada, Tao Zhao, Alan Litteneker, Abduallah Imran, and Ali Hatamizadeh, as well as my previous and senior colleagues, Sharath Gopal, Xiaowei Ding, Noah Duncan, Gregly Klar, Lap-Fai Yu, and Chenfanfu Jiang. It also was a pleasure to share our lab with some of Professor Joseph Teran's students, including Andre Prahadana, Qi Gao, and Daniel Ram. Even though we did not get the opportunity to collaborate on the same research projects, I learnt a lot from them. I would like to thank Professor Teran for his willingness to answer a non-trivial number of my questions.

My special thanks goes to the UCLA Computer Science Department, which financially supported me during my PhD program, through a departmental fellowship in my first year, followed by teaching assistantships. In the course of my teaching assistantships, I had the pleasure of working with Professor Paul Eggert, who impressed upon me the importance of teaching in the life of an academic professional. It was a real pleasure to contribute to the

education of all those undergraduate students. My teaching experiences and interactions with students taught me a lot more than is possible to learn within the confines of a research lab.

Finally, I would like to thank my parents, grandparents, and extended family for supporting me so far away from home. I would also like to thank my other friends in Los Angeles: Jaime Flor Flores, Nick Weires, Aviv Solodoch, Ran Gelles, Akshay Wadia, and Pratik Chaudari, who made my stay in LA so much more enjoyable. A very personal thank you goes to my partner, Monica, for her patience and support, especially during the stressful periods of my PhD program.

VITA

2011–2012	Software Engineer Datonics LLP Tel Aviv, Israel
2012–2013	Software Engineer Parametric Technology Corp. Herzeliya, Israel
2013	B.S. Computer Science Tel Aviv University Tel Aviv, Israel
Summer 2015	Software Engineer Intern Bloomberg LP New York City, NY
2015	M.S. Computer Science University of California, Los Angeles Los Angeles, CA
Summer 2016	Machine Learning Intern A9.com (Amazon) Palo Alto, CA
Summer 2017	Research Intern Autodesk Research San Fransisco, CA
2014–2018	Teaching Assistant Computer Science Department University of California, Los Angeles Los Angeles, CA

PUBLICATIONS

T. Weiss, A. Litteneker, N. Duncan, C. Jiang, L. Yu, D. Terzopoulos. "Fast, Scalable Layout Synthesis." Under review by the *IEEE Transactions on Visualization and Computer Graphics*, 2018.

M. Nakada, T. Zhou, H. Chen, T. Weiss, and D. Terzopoulos. "Deep Learning of Biomimetic Sensorimotor Control for Biomechanical Human Animation." In *ACM Transactions on Graphics*, **37**(4):1–14, August 2018. (*Proc. ACM SIGGRAPH 2018*, Vancouver, Canada, August 2018.)

T. Weiss, A. Litteneker, C. Jiang, D. Terzopoulos. "Position-Based Multi-Agent Dynamics for Real-Time Crowd Simulation". In *Proc. X International ACM SIGGRAPH Conference on Motion in Games (MIG '17)*, Barcelona, Spain, November 2017, pp. 10:1–10:8,
Best Paper Award.

T. Weiss, A. Litteneker, C. Jiang, D. Terzopoulos. "Position-Based Multi-Agent Dynamics for Real-Time Crowd Simulation"(Extended Abstract). In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '17)*, Los Angeles, USA, July 2017, pp. 27:1–27:2

T. Weiss, M. Nakada, and D. Terzopoulos. "Automated Layout Synthesis and Visualization From Images of Interior or Exterior Spaces." In *Proc. III IEEE Workshop on Vision Meets Cognition: Functionality, Physics, Intentionality, and Causality* , *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17)*, Honolulu, HI, July 2017, pp. 41–47.

T. Weiss, G. Klar, D. Terzopoulos. "Optimizing Design of Physical Objects for Fabrication." *UCLA Computer Science Department Technical Report No. 170005*, Los Angeles, USA, 2015.

CHAPTER 1

Introduction

Computer graphics researchers have long focused on modeling and simulating virtual worlds. Popular simulation topics include, for example, deformable solids (such as cloth), fluids, and human characters. Common concerns pertain to how to simulate all these phenomena realistically, in real-time if possible, and in a manner that scales tractably. Modeling refers to the process of creating these objects and the virtual worlds that they occupy (Figure 1.1). To model virtual worlds, researchers aspire to design tools such that the design process will be minimally manual and laborious. Hence, we are motivated to devise new tools and algorithms to facilitate the modeling process. These tools can be interactive, automatic, or semi-automated.

Many research problems in modeling and simulation may be formulated mathematically as optimization problems. In general, an optimization problem requires maximizing or minimizing an objective function. An optimization method seeks the value(s) of the variables in the objective function's domain that yield the maximum or minimum—i.e., optimum—value(s) of the function. Objective function optima may be local or global. A local optimum is at least as good as any nearby points in the domain, whereas a global optimum is at least as good as any other feasible point in the domain.

To solve an optimization problem effectively, we must choose an appropriate optimization algorithm. The choice depends on different factors, such as:

- Computational cost: What is the computational budget for a finding a feasible solution? Some applications, such as computer games, require a real-time, interactive response.

Figure 1.1: Typical setup for a 3D modeling artist.

- Optimum type: Does the problem require finding the global optimum, or would a local optimum suffice?
- Accuracy: How close to optimal need a "solution" be in order to satisfy the task at hand?

The choice of optimization algorithm depends on the complexity of the optimization landscape generated by the terms in the objective function (Figure 1.2).

Some optimization problems have a special structure that simplifies finding the optima. In the case of a convex optimization problem, a local optimum of the objective function is also the global optimum. For example, the optimal solution for a simple quadratic cost function

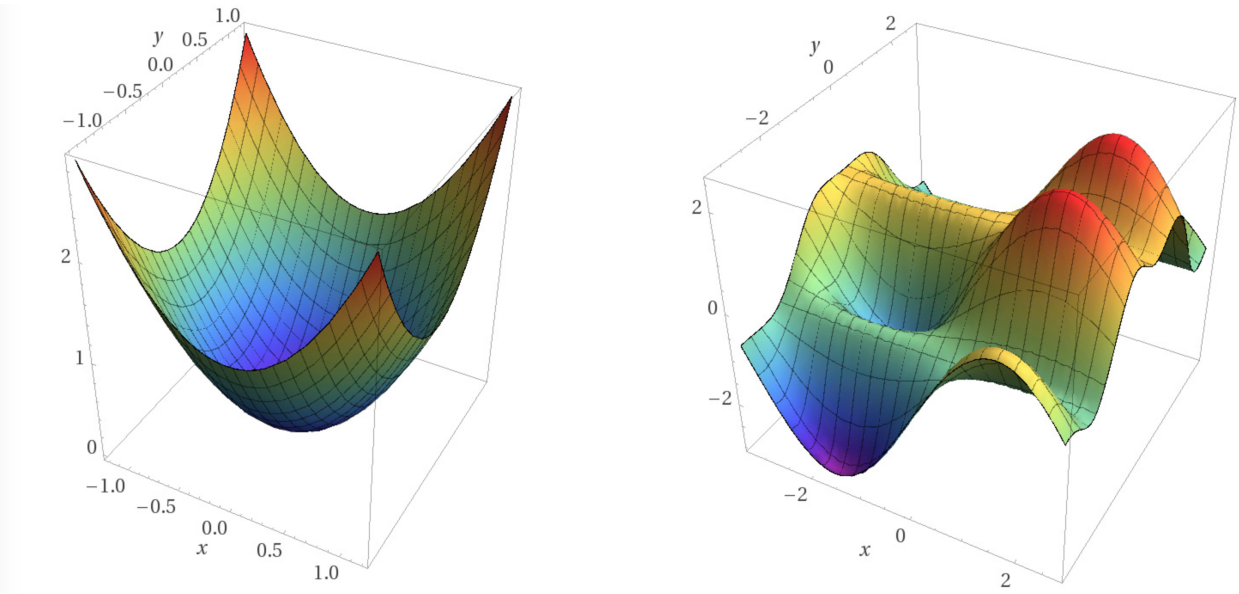


Figure 1.2: Illustration of a convex (left) vs nonconvex (right) objective function. A convex problem has a unique global optimum, while a nonconvex problem has multiple local optima.

can be found by the least-squares method. If the optimization landscape is highly nonconvex, employing least-squares will most likely not yield a satisfactory solution. Fortunately, stochastic optimization methods such as Metropolis-Hastings (Chib and Greenberg, 1995), simulated annealing (Kirkpatrick, 1984), genetic algorithms (Grosso, 1985), and others, are more likely to obtain optimal solutions. These methods are generally applicable to almost any type of optimization problem; however, the downside of stochastic approaches is that they are computationally expensive when confronting medium and larger-scale problems, which makes them inappropriate for most applications that require real-time interactivity and controllability.

Classical simulation techniques in graphics follow a Newtonian, force-based approach, in which forces drive the movements of the objects to be simulated in accordance with Newton's laws of motion. Based on the forces, and given the masses of the objects of interest, the velocities and positions of these objects are computed using numerical methods. However, graphics researchers have proposed alternative simulation techniques based on constraints (Witkin et al., 1987; Witkin and Kass, 1988). In a constraint-based simulation, each timestep

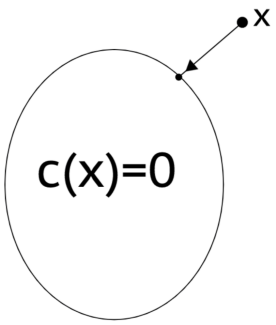


Figure 1.3: PBD satisfies constraints by finding the closest point on the constraint manifold.

in the simulation can be viewed as step towards solving a constrained optimization problem.

Viewing the simulation as an optimization is the essence of the so-called Position-Based Dynamics (PBD) approach, which was independently proposed by Müller et al. (2007) and later by Stam (2009). PBD has become increasingly popular for real-time simulation in games. Instead of computing forces, PBD directly computes positions in each time step, which often affords more direct control over the simulation. The computation of the positions is based on the approximate solution of an optimization problem, which we can interpret as finding the closest point on the constraint manifold (Figure 1.3). While this approach is approximate and not an accurate simulation of a physical system, it is satisfactory for games, for which visual plausibility suffices. Hence, PBD's popularity is driven by its simple formulation in terms of position-based constraints and its ability to produce visually satisfying simulations at interactive rates.

Surprisingly, little work has been done in generalizing PBD as an optimization-based framework for other graphics problems, such as layout modeling and crowd simulation. Modeling layouts is an important research topic in graphics and other domains such as interior design and architecture. When modeling layouts, we either create and refine virtual worlds, or design future real-world layouts. These layout are then used for other purposes, for example simulating crowds of people in a computer game. Crowd simulation in virtual environments has long occupied researchers in graphics and other research communities, such as urban studies. Algorithms that can simulate crowds in real-time are particularly

important for computer games, visual effects, and other entertainment applications. This thesis explores how to adapt PBD to these research goals, with the objective of achieving computational efficiency, stability, and speed advantages over previous work.

1.1 Contributions of the Thesis

This thesis develops new algorithms and software tools to help professionals and non-professionals in domains ranging from modeling and computational design to visual effects for gaming and simulation. In particular, we propose the following:

- A PBD approach to simulating real-time virtual crowds. This work has been published as (Weiss et al., 2017a).
- A PBD approach to the fast and scalable synthesis of layouts. This work has been published as (Weiss et al., 2018, 2017b).

1.1.1 Position-Based Multi-Agent Dynamics for Real-Time Crowd Simulation

The realistic simulation of crowds is important in several domains, from computer games and visual effects, to real-life situational analysis such as evacuation scenarios, urban planning, architectural simulation, and multi-agent planning and coordination. Thalmann et al. (2009) define a crowd simulation by the following criteria:

- To be qualified as a "crowd", a simulation should involve more than 1,000 virtual characters (Figure 1.4).
- The simulation should be interactive, running at a frame rate of 30 Hz. This frame rate is extremely important for some of the aforementioned use cases.
- A crowd should have a nice appearance, with individuality and variety (Figure 1.5). This variety can be expressed in the body types, or any other visually noticeable qualities of the virtual characters, such as an agent's actions, gait, walk and run cycles, etc.

Figure 1.4: A war scene in the computer game Medieval Total War 2, which contains more than a 1,000 agents.

- The crowd should be engaged in a specific scenario to be simulated, reacting to events, and having goals.

There are numerous problems to tackle in a crowd simulation task. The task can be divided to the following main components: modeling the crowd to be simulated, defining the scenario to be simulated, computing the movements of the characters, and rendering the computed motion of each character (Reynolds, 1999).

We address the task of computing the local movement of characters; i.e., in each time step, a character needs to decide in which direction to move. Usually, a character has a locomotion target; for example, a door in the end of a corridor. A typical scenario encompasses a scene with walls and obstacles, numerous virtual characters, and their respective locomotion goals. A successful crowd simulation method will have the characters make realistic progress towards their locomotion goal, while taking all these factors into account, including possible

Figure 1.5: Variety in crowd simulations. Characters have different hair styles, clothing, accessories, among other visual qualities.

path planning and collisions between the characters.

The goal of this thesis is to develop a method that is capable of simulating large to huge crowds in real time (Figure 1.6). Since alternative approaches are expensive, we approximate the crowd using a particle-based model. Particle-based crowd models are common in multiple research communities, such as graphics, physics, and computational social science. In the particle-based approach, forces control the behavior of the characters (a.k.a., agents), pushing them toward their locomotion goal, and acting locally to avoid collisions. While carefully designed models, such as the social force model (Helbing and Molnar, 1995) and the power law model (Karamouzas et al., 2014), can yield some realistic crowd behaviors, they occasionally require elaborate numerical treatments to remain stable and robust. An additional challenge stems from the difficulty in leveraging all these competing forces into a unified model that scales and works well for both sparse and dense crowds. Furthermore, in most cases, the agents to be simulated represent humans, so a particle model has to take certain physical constraints into account when computing the resulting motions in each time

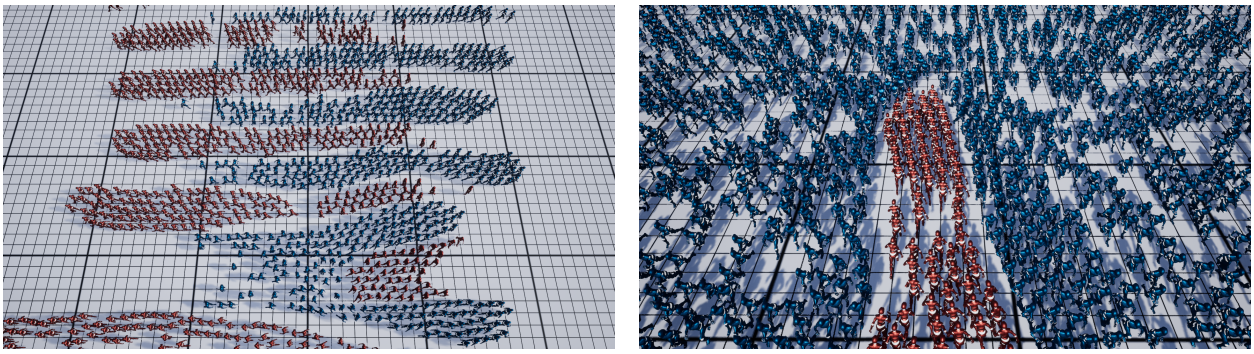


Figure 1.6: Our PBD-based crowd simulation method animates both sparse and dense groups of agents at interactive rates.

step; for example, not allowing agents to locomote a distance that is more than humanly possible, not allowing extreme velocities, accelerations, changes of orientation, etc.

In this thesis, we investigate PBD as an alternative approach for simulating both dense and sparse crowds. Given the success of PBD in simulating various solid and fluid materials in real-time physics, our work further extends the concept to crowd simulation. Our motivation is to develop a numerical framework for crowd simulation that is robust, stable, and easy to implement, ideally for use in interactive games. Due to the flexibility of PBD in defining positional constraints among particles, our proposed framework provides a new platform for artistic design and control of agent behaviors in crowd modeling and animation. Furthermore, we adopt the PBD approach since it is an unconditionally stable scheme. Even though it may not always converge accurately to the solution manifold, a nonlinear Gauss-Seidel-like constraint projection enables the algorithm to produce satisfactory results with modest computational cost suitable for real-time applications. Additionally, the resulting solution scheme is easy to implement and does not require any linear solves.

The method that we propose in this thesis can simulate both sparse and dense crowds comprised of up to 100,000 or more agents, in real time. Furthermore, this method is easily implementable for anyone using existing PBD software. Our work makes following technical contributions:

- We show how crowds can be simulated within the PBD framework by augmenting it

with a non-passive agent-based planning velocity.

- We adopt the position-based frictional contact constraints of granular materials to model local collision avoidance among nearby agents. An XSPH viscosity term is also added to approximate coherent and collective group behavior.

- We develop a novel long-range collision avoidance constraint to deal with anticipatory collisions. Our model permits the natural development of agent groups.

- We demonstrate multi-species crowd coupling by supporting spatially varying Lagrangian physical properties.

1.1.2 Fast, Scalable Layout Synthesis

The aesthetic and creative process of interior design is laborious and surprisingly complex; hence, it is usually done by professional interior layout designers with extensive training. For example, to find a desirable furniture arrangement 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. It would be helpful to have software that can aid professionals or less experienced users in this design process.

Typically, a realistic indoor scene is populated with different kinds of furniture items and accessories (Figure 1.7). All these pieces of furniture should be placed according to different interior design constraints, some of which are conflicting. Interior layout synthesis research aims to automatically generate such a layout given these constraints, for both real and virtual worlds. Layouts in virtual worlds are useful for game design, educational, and training purposes, and are more useful in practice if they are larger, more complex, realistic and can be quickly synthesized autonomously.

In recent years, researchers have proposed several methods for synthesizing layouts, either automatically or interactively (Smelik et al., 2014). Some of these methods pose layout synthesis as a highly nonconvex optimization problem with many complex constraints. The method proposed by Yu et al. (2011) is considered the baseline comparison for future auto-

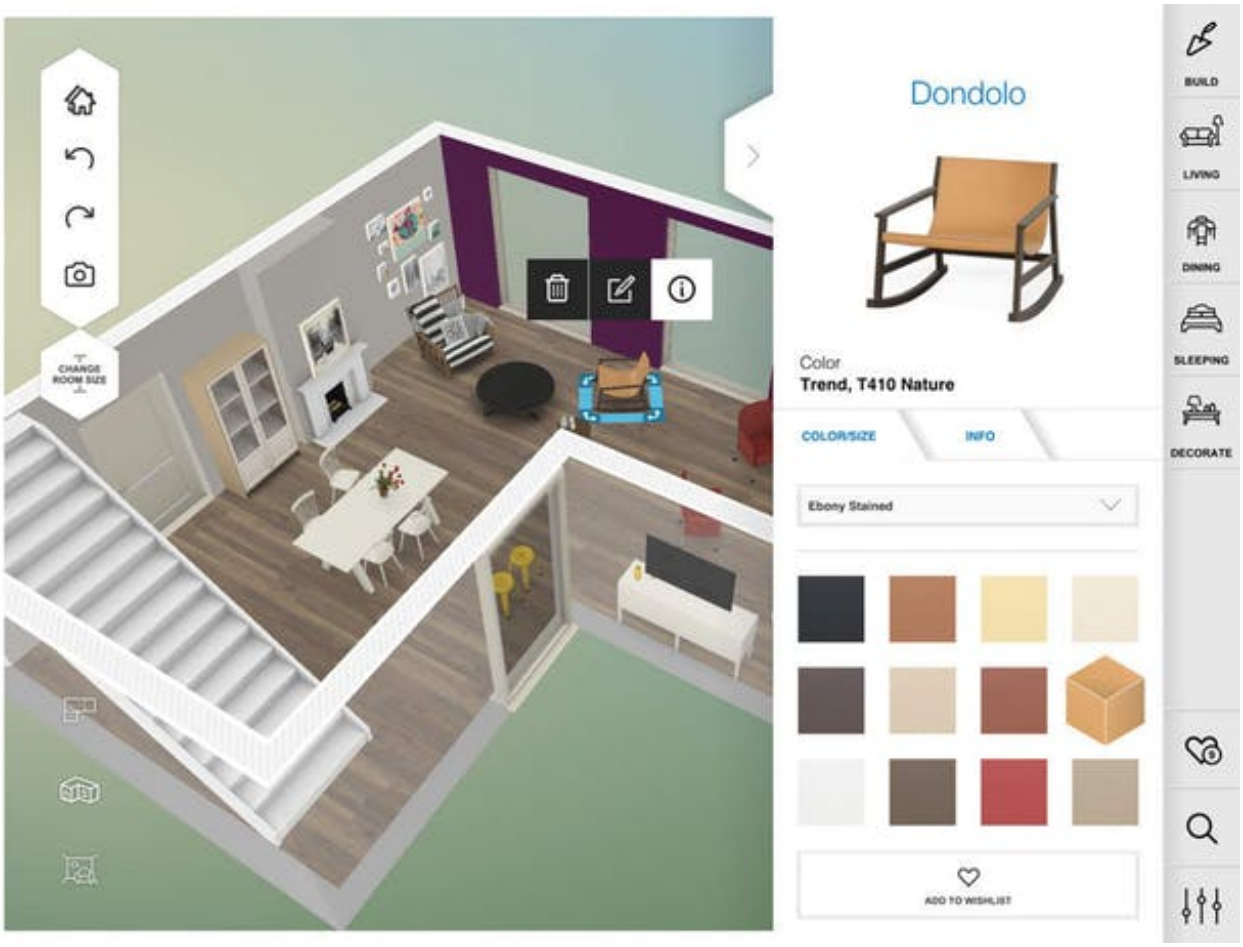


Figure 1.7: Home design software is now available for users with mobile phones. For example, *Amikasa*, an interior layout and room modeling software available on iOS and Android.

matic layout synthesis work. Due to the complex nature of the layout synthesis problem, and since constraints are often difficult to express as differentiable functions, Yu et al. apply a probabilistic scheme to effectively sample viable layout candidates. The underlying machinery for this sampling is based on Markov chain Monte Carlo (McMC) (Chib and Greenberg, 1995). While this work and similar approaches (Merrell et al., 2011; Yeh et al., 2012) are able to obtain satisfactory results for a mélange of different layout considerations, they do not scale to layouts containing more than a few objects, due to the high computational costs involved in the underlying McMC machinery, resulting in very long run times that are unsuitable for interactive purposes. Also, when confronted with a large numbers of objects, the majority of these techniques become intractable because of the high dimensional search

spaces.

To overcome these issues, we propose a novel continuous framework for layout synthesis, which is interactive for most interior layouts, and also seamlessly scales to layouts that contain up to hundreds of objects. Our main observation is that there are a lot of common factors between elasticity-inspired simulation of deformable objects and layout synthesis. Elasticity penalizes deformation of a model and layout synthesis penalizes constraint violation; i.e., elasticity pulls a deformed model towards a goal configuration, the same way a layout enforces its geometric layout constraints. Both can be formulated as optimization problems amenable to the PBD approach. Consequently, our PBD-based method enables the fast generation of dense, large-scale layouts that are intractable using previous approaches. Like probabilistic methods, however, our continuous approach can synthesize multiple viable layouts in a given environment. To our knowledge, this is the first time a physics-based approach has been applied to layout synthesis.

The input for our method is an empty environment, a number of layout objects, and a set of layout constraints. Initially the orientations and positions of the objects are randomized, which is analogous to choosing a random initial guess in an iterative solver. It then iteratively modifies the positions and orientations to achieve viable layouts (Figure 1.8). At each iteration, the objects are moved so as to satisfy competing constraints. We stop the iterative solver when the synthesized layout satisfies most constraints and the layout quality does not improve.

Our main contributions are as follows:

1. We propose a novel, physics-based framework for layout synthesis with a PBD-based solution.

2. We formulate a set of common layout constraints within our framework.

3. We demonstrate that our approach is faster than previous layout synthesis work by an order of magnitude, enabling the quick prototyping complex layouts.

Figure 1.8: A tightly-packed picnic layout (top), and a theater layout with a large number of chairs (bottom), automatically placed by our method given user-specified constraints that include distance, viewing angle, and spaciousness criteria.

1.2 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2 reviews relevant related work.

Chapter 3 provides an overview of the position-based dynamics technique.

Chapter 4 presents our novel, PBD-based crowd simulation method.

Chapter 5 tackles the problem of interior layout synthesis with our PBD-based approach.

Chapter 6 reports on experiments and results with our two methods.

Chapter 7 presents our conclusions and proposes avenues for future work.

Appendix C presents an augmented reality application of our layout synthesis technique.

CHAPTER 2

Review of Related Work

In this chapter, we give a brief overview of research related to each of the two problems that we address in this thesis.

2.1 Physics-Based Modeling

Physical simulation is a well-studied problem in the computational sciences. Physics-based modeling techniques have been used in various contexts, from animation (Terzopoulos et al., 1987; Manteaux et al., 2016), to geometric design (Qin and Terzopoulos, 1996; Attar et al., 2009), to architectural floor plan design (Harada et al., 1995; Arvin and House, 2002). Traditional simulation methods include the Finite Difference Method (Terzopoulos et al., 1987), the Finite Element Method (FEM) (O'brien and Hodgins, 1999), the Finite Volume Method (Teran et al., 2003), the boundary element method (James and Pai, 1999), and particle-based approaches (Desbrun et al., 1999). Graphics researchers have adopted these methods for animation, with the main goal of recreating physical effects in entertainment industry applications, such as visual effects in motion pictures and computer games. Hence, computational time and interactivity are important factors in addition to visual plausibility. This is especially true for real-time applications.

2.2 Position-Based Methods

Position-Based Dynamics (PBD), independently introduced by Müller et al. (2007) and later by Stam (2009), was proposed to simulate physical models in situations where the simulation

speed and robustness takes priority over physical realism.[1] The method works by iteratively projecting the solution state onto a set of constraints (see Section 3 for more details).

Later, Macklin et al. (2014a) expended this early work, and presented a unified PBD solver for various natural phenomena, such as fluids, smoke, and granular materials. Recently, XPBD was proposed to eliminate the iteration count and time step dependence of PBD (Macklin et al., 2016). Even though PBD traditionally defines geometric constraints among particles, it can also approximate force responses from continuum mechanics. Bender et al. (2014a) formulated continuum energies as PBD constraints. Müller et al. (2014) defined constraints based on element strain measures. The close relationship between PBD and popular continuum-mechanics-based discretization was further explored in recent work on optimization-based methods for real-time animation (Liu et al., 2013; Bouaziz et al., 2014a; Wang, 2015; Narain et al., 2016).

Bouaziz et al. (2014b) recently introduced projective dynamics, a point-based method that is a more physically principled modification to PBD. The method was further generalized by Narain et al. (2016). Tonge et al. (2012) used a Jacobi-based method for the large-scale simulation of rigid-body dynamics. In the Jacobi method, changes to the states of the objects, subject to the constraints, are computed in parallel rather than sequentially as in the Gauss-Seidel method.

A more complete survey of PBD is provided by Bender et al. (2014b).

2.3 Crowd Simulation

There exist two main simulation models for computing the movement of crowds of agents: The artificial life approach (Shao and Terzopoulos, 2007; Yu and Terzopoulos, 2007), and particle-based models (Karamouzas et al., 2014; Helbing et al., 2000, 2007). In this thesis, we pursue the latter.

[1] PBD is part of a larger family of physical simulation methods called point-based methods. The common property of these methods is that they do not explicitly compute physical quantities such as momentum and forces, but instead work directly with positions. Bender et al. (2014b) present a survey.

Out of the various simulation considerations above, we focus on collision avoidance between crowd agents. Collision avoidance algorithms can be classified into discrete and continuum approaches (Golas et al., 2013). Continuum approaches, such as the technique proposed by Narain et al. (2009) have proven efficient for large-scale dense crowds, but are less suitable for sparse crowds. Force-based discrete approaches, such as the recently proposed power-law model (Karamouzas et al., 2014), are well suited for sparse crowds, but can be computationally expensive and may require smaller time steps due to explicit time integration.

Efficient, natural, and stable collision avoidance in sparse and dense distributions of agents remains a very active area of research in crowd simulation. In multi-agent simulations, it is essential to capture both individual local behaviors and aggregate collective behaviors. Continuum approaches—such as 'Continuum Crowds' (Treuille et al., 2006), a crowd model that uses continuum dynamics to simulate pedestrian flow—are particularly suitable for dense crowds and complex environments (Jiang et al., 2010). Unfortunately, the traditional regime of pure continuum models tends to smooth out local agent behaviors, motivating research on hybrid methods. For example, Narain et al. (2009) simulated dense crowds with a hybrid, Eulerian-Lagrangian particle-in-cell approach and the unilateral incompressibility constraint (UIC), which has proven to be an effective assumption for crowds. Subsequently, frictional forces were taken into account in modeling crowd turbulence (Helbing et al., 2007; Golas et al., 2014), which is essential in extra high-density scenarios. This has also inspired us to treat dense agent collisions with a frictional contact model similar to PBD dry sand simulation (Macklin et al., 2014a). To robustly model multiple densities, Golas et al. (2013) proposed a hybrid scheme for simulating high-density and low-density crowds with seamless transitions.

Other techniques for collision avoidance have been proposed. Many researchers adopted force-based models (Reynolds, 1987, 1999; Helbing et al., 2000). An interaction energy between pedestrians was modeled with a power law in recent and concurrent work by Karamouzas et al. (2014, 2017). As an alternative to forces, the reciprocal velocity obstacle was proposed in robotics for multi-agent navigation (Van den Berg et al., 2008). Guy et al. (2009) extended velocity obstacles and used a parallel optimization framework for col-

lision avoidance. Ren et al. (2016b) augment velocity obstacles with velocity connections to keep agents moving together, thus allowing more coherent behaviors. He et al. (2016) simulated dynamic group behaviors based on the least effort principle. Guy et al. (2010a) simulated large-scale crowds by optimization based on the Principle of Least Effort. Bruneau and Pettré (2015) presented a mid-term planning system to fill in the gap between long-term planning and short-term collision avoidance.

2.4 Layout Synthesis

We focus on layout synthesis problems in which a set of objects is to be arranged in an open space. The objects are assumed to be rigid bodies. The goal of the layout problem is to position and orient the objects such that they satisfy several functional and aesthetic criteria. These criteria are encoded in the form of a complex, nonconvex objective function that typically has multiple terms. The main challenge stems from finding an arrangement that respects conflicting terms, resulting in a multitude of possible layout outcomes, some of which may be unsatisfactory. There are numerous publications in this category. For brevity, we focus on a few related ones:

Yu et al. (2011) and Merrell et al. (2011) introduced an McMC-based approach for furniture arrangement. Yeh et al. (2012) formulated layout constraints with factor graphs, allowing a variable number of elements in the synthesized layout. These McMC-based, probabilistic sampling methods can synthesize scenes that respect a complex and conflicting set of constraints, but they have been shown to work only on layouts with a limited number of objects and relaxed spacing. The underlying inefficiency of these approaches stems from the fact that they do not employ local gradient information—they merely sample a new position of a furniture item by applying a shift move to the current position.

Feng et al. (2016) optimized mid-scale layouts by stochastically optimizing an objective function derived from agent-based simulation. Fisher et al. (2012) generated small, local layouts of objects in a scene, guided by exemplars—e.g., the layout of items on a desk. Layouts are generated by sampling probabilistically from an occurrence model distribution.

Additionally, the authors report that since the layout generation is probabilistic, it cannot handle hard constraints such as rigid grid layouts or exact alignment relationships.

Peng et al. (2014) introduced a method that creates a layout out of deformable templates, which differs from prior work that assumed the objects were rigid. They incorporated a continuous method in their approach, but use it only to deform objects, whereas we use ours for positioning objects.

Layout research has also focused on domains other than interior design. Majerowicz et al. (2014) focused on adding objects to shelves in a 2D setting. Bao et al. (2013) use a combined stochastic and numerical optimization approach to explore and refine building layouts. Zhu et al. (2012) synthesized layouts of mechanical components to control the motion of a toy. Cao et al. (2012, 2014) synthesized manga layouts. Reinert et al. (2013) introduced an interactive layout generation method for arranging shapes according to aesthetic attributes, such as color and size. In this thesis, we confine ourselves to interior design layouts, but our method generalizes to other domains as well.

CHAPTER 3

Review of Position-Based Dynamics

Traditional physics-based simulation approaches in computer graphics are force-based. Internal and external forces are accumulated, from which accelerations are computed based on Newton's second law of motion. A time integration method is then used to update the velocities, and finally the positions of the object.

By contrast, position-based approaches do not explicitly compute physical quantities such as forces. Rather, these methods directly compute the positions of the objects based on predefined positional constraints. Simulating a scenario based on these positional constraints leads to an approximate solution, which is not always physically accurate. However, the advantage of position-based approaches is that they provide a high level of control and are stable even for simple time integration schemes. Due to their simplicity, robustness and speed, these approaches have recently become popular in computer graphics, particularly in the interactive environments of computer games where accurate physical simulations are typically unnecessary.

The remainder of this chapter overviews Position-Based Dynamics (PBD).

3.1 Fundamentals

Position-based dynamics simulates a scenario by directly modifying the positions of the objects themselves. The simulation objects are represented by a set of points and a set of constraints between these points. For example, a rope can be subdivided to multiple points with distance constraints imposed between the points (Figure 3.1).

These constraints are defined by a constraint function (Baraff and Witkin, 1998; Teschner

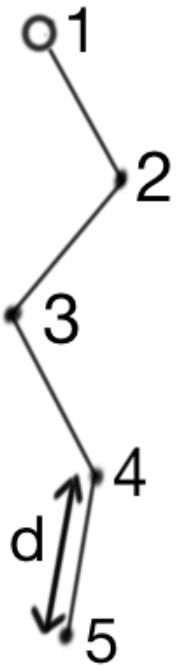


Figure 3.1: A rope represented by points with distance constraints between them.

et al., 2004). Instead of computing forces as the derivative of a constraint function energy, as in traditional approaches, PBD directly solves for the equilibrium configuration and projects positions.

Many constraints are possible in the PBD framework. In this chapter, we discuss the distance constraint and its derivation. Unless otherwise stated in the following chapters, other PBD constraints are out of the scope of this thesis (refer to (Bender et al., 2014b) for a complete survey). A distance constraint preserves a distance between two points during a simulation timestep. These points might have moved because of external forces, collisions, or other physics-based effects. Given a distance constraint $C(\boldsymbol{x})$ that is violated, we want to find a positional correction so that $C(\boldsymbol{x} + \Delta\boldsymbol{x}) = 0$.

PBD enforces the positional correction $\Delta\boldsymbol{x}$ to be in the direction of the gradient of the constraint:

$$C(\boldsymbol{x} + \Delta\boldsymbol{x}) \approx C(\boldsymbol{x}) + \nabla_x C(\boldsymbol{x}) \cdot \Delta\boldsymbol{x} = 0. \tag{3.1}$$

The correction is restricted to be in the direction of the gradient:

$$\Delta \boldsymbol{x} = \lambda \nabla_x C(\boldsymbol{x}), \tag{3.2}$$

where λ is a scaler for which the equality holds. Defining $s = -\lambda$, substituting (3.2) into (3.1), and solving for s, yields

$$s = \frac{C(\boldsymbol{x})}{||\nabla_x C(\boldsymbol{x})||^2},$$

and consequently

$$\Delta \boldsymbol{x} = -\frac{C(\boldsymbol{x})}{||\nabla_x C(\boldsymbol{x})||^2} \nabla_x C(\boldsymbol{x}). \tag{3.3}$$

This generalizes for constraints with n points, so that

$$\begin{aligned} s =& \frac{C(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)}{\sum_i ||\nabla_{x_i} C(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)||^2}; \\ \Delta \boldsymbol{x}_i =& -\frac{C(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)}{\sum_i ||\nabla_{x_i} C(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)||^2} \nabla_{x_i} C(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n). \end{aligned}$$

In case each point $\boldsymbol{x}_i$ has a different mass m_i, we weight the corrections $\Delta \boldsymbol{x}$ by the inverse masses $w_i = \frac{1}{m}$. Replacing (3.2) by

$$\Delta \boldsymbol{x}_i = -s w_i \nabla_{x_i} C(\boldsymbol{x}), . \tag{3.4}$$

And the scaling factor becomes

$$s = \frac{C(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)}{\sum_i w_i ||\nabla_{x_i} C(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)||^2} \nabla_{x_i} C(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n). \tag{3.5}$$

We now illustrate the derived corrections on a distance constraint between 2 points: $C(\boldsymbol{x}_1, \boldsymbol{x}_2) =$

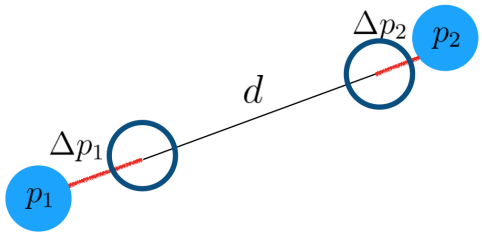


Figure 3.2: Two point p_1 and p_2, constrained to be at a distance d, which is satisfied by the positional corrections of Δp_1 and Δp_2.

$|\boldsymbol{x}_1 - \boldsymbol{x}_2| - d = 0$. The derivatives with respect to the points are

$$\begin{aligned}\nabla_{x_1} C(\boldsymbol{x}_1, \boldsymbol{x}_2) =& \frac{\boldsymbol{x}_1 - \boldsymbol{x}_2}{|\boldsymbol{x}_1 - \boldsymbol{x}_2|} \\ \nabla_{x_2} C(\boldsymbol{x}_1, \boldsymbol{x}_2) =& -\nabla_{x_1} C(\boldsymbol{x}_1, \boldsymbol{x}_2),\end{aligned}$$

and the scaling factor is

$$s = \frac{|||\boldsymbol{x}_1 - \boldsymbol{x}_2|| - d|}{w_1 + w_2},$$

so that the positional corrections are

$$\begin{aligned}\Delta \boldsymbol{x}_1 =& s w_1 \nabla_{x_1} C(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\boldsymbol{x}_1 - \boldsymbol{x}_2}{|\boldsymbol{x}_1 - \boldsymbol{x}_2|} \frac{|||\boldsymbol{x}_1 - \boldsymbol{x}_2|| - d|}{w_1 + w_2} \\ \Delta \boldsymbol{x}_2 =& s w_2 \nabla_{x_2} C(\boldsymbol{x}_1, \boldsymbol{x}_2) = -\frac{\boldsymbol{x}_1 - \boldsymbol{x}_2}{|\boldsymbol{x}_1 - \boldsymbol{x}_2|} \frac{|||\boldsymbol{x}_1 - \boldsymbol{x}_2|| - d|}{w_1 + w_2}.\end{aligned}$$

These corrections are illustrated in Figure 3.2:

While the above positional corrections fully solve the constraint, there are particular simulation scenarios in which that is not desirable; e.g., in the case of elastic material that does not immediately return back to its former shape. To that end, PBD simulates elasticity by incorporating a stiffness parameter $k \in [0, 1]$, which multiples the positional correction in (3.4), such that $\boldsymbol{x}_i = -skw_i \nabla_{x_i} C(\boldsymbol{x})$. Then, since there may be numerous iterations in a PBD timestep, k is adjusted: $k = 1 - (1 - k)^{\frac{1}{n_s}}$. Note that this adjustment has a

Algorithm 1 General PBD Solver

1: **for all** point i **do**
2: Update velocity $\boldsymbol{v}_i^{n+1} \leftarrow \boldsymbol{v}_i^n + \Delta t f_{ext}(\boldsymbol{x}_i)$
3: Update predicted position $\boldsymbol{x}_i^* \leftarrow \boldsymbol{x}_i^n + \Delta t \boldsymbol{v}_i^{n+1}$
4: **end for**
5: **while** iteration count< max iterations **do**
6: **for all** point i **do**
7: compute position correction $\Delta \boldsymbol{x}_i$
8: $\boldsymbol{x}_i^* \leftarrow \boldsymbol{x}_i^* + \Delta \boldsymbol{x}_i$
9: **end for**
10: **end while**
11: **for all** point i **do**
12: $\boldsymbol{v}_i^{n+1} \leftarrow (\boldsymbol{x}_i^* - \boldsymbol{x}_i^n)/\Delta t$
13: $\boldsymbol{x}_i^{n+1} \leftarrow \boldsymbol{x}_i^*$
14: **end for**

non-trivial affect on the stiffness of the constraint, and requires tuning according to the scenario simulated and the number of solver iterations. Luckily, real-time environments have a fixed number of solver iterations per time time step, so this dependency and tuning is not problematic. Recently, a different type of PBD scaler was proposed that allows for the stiffness not to depend on the number of iterations (Macklin et al., 2016). However, we shall not discuss this further, since this thesis does not consider methods for the simulation of elastic and other similar materials. We use PBD stiffness in a manner similar to weights in a multi-objective optimization problem.

3.2 Iterative Solver

Given a set of points $\boldsymbol{x}$, constraint functions acting on these points, and a time step Δt, we want to compute the positions of these points in the next time step (Algorithm 1).

$\boldsymbol{x}_i^n$ is the known position of point i at the beginning of the n-th timestep, and $\boldsymbol{x}_i^{n+1}$ is the point's unknown position in the next timestep. In each timestep, we first calculate the velocity of a point $\boldsymbol{v}_i^{n+1}$ considering external forces f_{ext} (line 2). $\boldsymbol{v}_i^n$ is the point's initial velocity, without the affect of these forces. Then, we update the points predicted position $\boldsymbol{x}_i^*$. This is the point's predicted position before any of the constraint functions are taken into

account. In the main solver loop (lines 5-10), we compute the affects of these constraints on the points predicted position $\boldsymbol{x}_i^*$. If point i is constrained, the constraint function contributes a positional correction. In case there are n constraints acting on point i, each constraint function contributes a correction that is then averaged: $\Delta x_i = \frac{1}{n}\sum_{j=1}^{n}\Delta x_i^j$. However, in some cases this averaging is too aggressive and the number of iterations required to reach a solution increases. To address this, a global user-defined parameter w can be introduced to control the rate of successive over-relaxation (SOR) (Golub and Varga, 1961):

$$\Delta x_i = \frac{\omega}{n}\sum_{j=1}^{n}\Delta x_i^j, \tag{3.6}$$

where $1 \leq \omega \leq 2$ is preferred by PBD researchers. Macklin et al. (2014b) propose $\omega = 1.2$, and we use the same in Chapter 4, but this choice my vary by the scene being simulated. After calculating the position correction, the predicted position of point i is updated (Line 8). The loop continues till we reach the maximum number of iterations (Line 7. Finally, the velocity of point i in the current timestep is updated based on the current predicted position (Line 12), and the initial position in the next time step is set to the current predicted position (line 13).

Since all the constraints are solved independently, the process is easily parallelizable. This solver is known as a Jacobi-style constraint solver. There are alternative approaches, however. One approach, for example, is to have each constraint update the current predicted position, and then have this updated predicted position visible to the next constraint acting on the same point. This is known as a Gauss-Seidel iteration. Fratarcangeli and Pellacini (2015) proposed a graph coloring approach, where the graph is modified such that it produces a desired number of k colors by splitting high valence points. These points are solved using the Jacobi approach. Another hybrid approach is to solve the first $k-1$ colors using $k-1$ Gauss-Seidel passes and then solve the remaining constraints with one Jacobi pass.

CHAPTER 4

Crowd Simulation

Exploiting the efficiency and stability of Position-Based Dynamics (PBD), we develop in this chapter a novel crowd simulation method that runs at interactive rates for hundreds of thousands of agents. Our method enables the detailed modeling of per-agent behavior in a Lagrangian formulation. We model short-range and long-range collision avoidance to simulate both sparse and dense crowds. On the particles representing agents, we formulate a set of positional constraints that can be readily integrated into a standard PBD solver. We augment the tentative particle motions with planning velocities to determine the preferred velocities of agents, and project the positions onto the constraint manifold to eliminate colliding configurations. The local short-range interaction is represented with collision and frictional contact between agents, as in the discrete simulation of granular materials. We incorporate a cohesion model for modeling collective behaviors and propose a new constraint for dealing with potential future collisions.

Section 4.1 overviews our algorithmic approach, and Section 4.2 discusses the algorithmic details and details the design of constraints.

4.1 Algorithm Overview

Our simulation loop per time step is similar to that for PBD, with several modifications. We outline our procedure in Algorithm 2 and highlight the different steps.

Assume that we have N agents. Each agent i, where $i = 1, 2, \ldots, N$, is represented with a fixed-sized particle with position $\boldsymbol{x}_i \in \mathbb{R}^2$ and velocity $\boldsymbol{v}_i \in \mathbb{R}^2$. For multi-species considerations, we treat each particle as a circle with radius r_i and mass m_i. When we are

Algorithm 2 Position-based crowd simulation loop.

for all agent i **do** ▷ §4.2.1
    calculate $\boldsymbol{v}_i^p$ from a velocity planner
    calculate a blending velocity $\boldsymbol{v}_i^b$ from $\boldsymbol{v}_i^p$ and $\boldsymbol{v}_i^n$
    $\boldsymbol{x}_i^* \leftarrow \boldsymbol{x}_i^n + \Delta t \boldsymbol{v}_i^b$
end for
for all agent i **do**
    find neighboring agents $S_i = \{s_{i1}, s_{i2}, \ldots, s_{im}\}$
end for
while iteration count< max stability iterations **do**
    for all agent i **do**
        compute position correction $\Delta \boldsymbol{x}_i$ ▷ §4.2.2
        $\boldsymbol{x}_i^n \leftarrow \boldsymbol{x}_i^n + \Delta \boldsymbol{x}_i$
        $\boldsymbol{x}_i^* \leftarrow \boldsymbol{x}_i^* + \Delta \boldsymbol{x}_i$
    end for
end while
while iteration count< max iterations **do**
    for all agent i **do**
        compute position correction $\Delta \boldsymbol{x}_i$ ▷ §4.2.2, §4.2.4, §4.2.5
        $\boldsymbol{x}_i^* \leftarrow \boldsymbol{x}_i^* + \Delta \boldsymbol{x}_i$
    end for
end while
for all agent i **do**
    $\boldsymbol{v}_i^{n+1} \leftarrow (\boldsymbol{x}_i^* - \boldsymbol{x}_i^n)/\Delta t$
    Add XSPH viscosity to $\boldsymbol{v}_i^{n+1}$ ▷ §4.2.3
    Clamp $\boldsymbol{v}_i^{n+1}$ ▷ §4.2.6
    $\boldsymbol{x}_i^{n+1} \leftarrow \boldsymbol{x}_i^*$
end for

stepping from time n to time $n+1$ in a traditional PBD simulation loop for passive physical simulations, a forward Euler position prediction is first performed as $\boldsymbol{x}_i^* = \boldsymbol{x}_i^n + \Delta t(\boldsymbol{v}_i^n + \Delta t \boldsymbol{f}_{\text{ext}}(\boldsymbol{x}_i^n))$, where $\boldsymbol{f}_{\text{ext}}$ represents external forces such as gravity. In position-based crowds, $\boldsymbol{x}_i^*$ needs to be computed differently to take into account the velocity planning of each agent. In particular, we compute $\boldsymbol{x}_i^*$ based on a blending scheme between a preferred velocity and the current velocity $\boldsymbol{v}_i^n$ (see Section 4.2.1). It is apparent that the predicted $\boldsymbol{x}_i^*$ for a particle completely ignores the existence of any other particles and just passively advects in the velocity field. To resolve this, PBD defines constraint functions on the desired location of the particles. Both equality and inequality constraints are supported, and they can be expressed as $C_k(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N) = 0$ and $C_k(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N) \geq 0$ respectively. Hence, the task is to

search for a correction $\Delta \boldsymbol{x}_i$ such that $\boldsymbol{x}_i^{n+1} = \boldsymbol{x}_i^* + \Delta \boldsymbol{x}_i$ satisfies the constraints. Once the new positions are computed, agent velocities can be updated as $\boldsymbol{v}_i^{n+1} = (\boldsymbol{x}_i^{n+1} - \boldsymbol{x}_i^n)/\Delta t$. This update guarantees stable agent velocities as long as the constraint projection is stable.

4.2 Details of the Algorithm

Our position-based formulation includes several modifications to the standard PBD scheme as well as additional constraints for short-range and long-range collision avoidance between agents.

4.2.1 Velocity Blending

Agent level roadmap velocity planning describes high-level agent behaviors. Local behavior may be influenced by factors such as social or cognitive goals, while global behavior may be specified by a particular walking path. We note that the roadmap planning is an orthogonal component to our constraint-based scheme.

In the physics-based simulation of solids and fluids, particles generally retain their existing velocities. In particular, as demonstrated by Bouaziz et al. (2014a), the implicit Euler time integration of a physical system can be formulated as an minimization problem that balances the 'momentum potential' $\|\boldsymbol{M}^{1/2}(\boldsymbol{x} - (\boldsymbol{x}^n + \Delta t \boldsymbol{v}^n))\|_F^2/2\Delta t^2$ and other potential energies, where $\boldsymbol{M}$ is the mass matrix. In a multi-agent crowd simulation, it is similarly more desirable to include the inertia effect before predicting an agent's desired velocity. Denoting the preferred velocity given the planner with $\boldsymbol{v}_i^p$, we calculate the agent velocity $\boldsymbol{v}_i^b$ as a linear blending between $\boldsymbol{v}_i^p$ and the current velocity $\boldsymbol{v}_i^n$, as follows:

$$\boldsymbol{v}_i^b = (1 - \alpha)\boldsymbol{v}_i^n + \alpha \boldsymbol{v}_i^p, \tag{4.1}$$

where $\alpha \in [0, 1]$. We set $\alpha = 0.0385$ in all our simulations. A more adaptive choice, such as the density-based blending factor as in (Narain et al., 2009), can also be used in our framework.

4.2.2 Frictional Contact

We model local particle contacts with an inequality distance constraint as in standard position-based methods:

$$C(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| - (r_i + r_j) \geq 0, \tag{4.2}$$

where r_i and r_j are the radii of agents i and j. To model frictional behavior between neighboring agents, we further adopt kinematic frictions as described by Macklin et al. (2014a).

4.2.3 Cohesion

To encourage more coherent agent motions, we add the artificial XSPH viscosity (Schechter and Bridson, 2012; Macklin and Müller, 2013) to the updated agent velocities. Specifically,

$$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + c \sum_j (\boldsymbol{v}_i - \boldsymbol{v}_j) W(\boldsymbol{x}_i - \boldsymbol{x}_j, h), \tag{4.3}$$

where $W(\boldsymbol{r}, h)$ is the Poly6 kernel for SPH (Macklin and Müller, 2013). For our simulations, with particles with radius 1, we use $h = 7$ and $c = 217$.

4.2.4 Long Range Collision

Karamouzas et al. (2014) describe an explicit force-based scheme for modeling crowds. We design a similar scheme as a position-based constraint. As in their power law setting, the leading term is the time to collision $\boldsymbol{\tau}$, defined as the time when two disks representing particles i and j touch each other in the future. As in (Karamouzas et al., 2014), it can be shown that

$$\tau = \frac{b - \sqrt{b^2 - ac}}{a}, \tag{4.4}$$

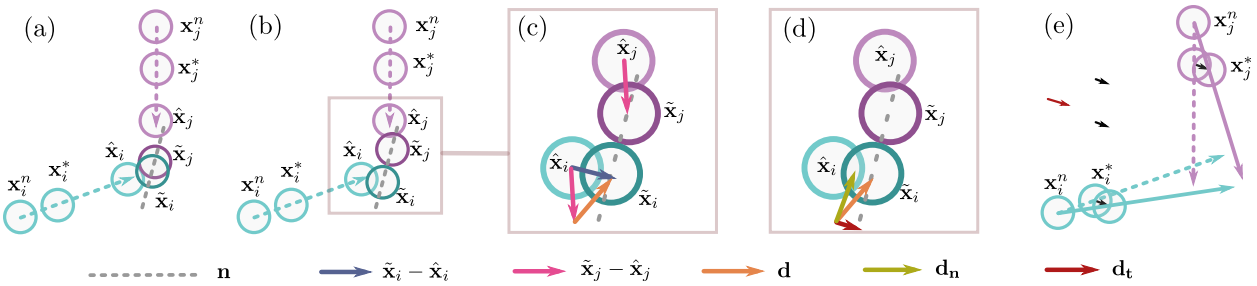


Figure 4.1: Avoidance model for predictive collision avoidance. (a) Starting with particles in current positions $\boldsymbol{x}_i^n$ and $\boldsymbol{x}_j^n$, PBD estimates their positions $\boldsymbol{x}_i^*$ and $\boldsymbol{x}_j^*$ at the next time step. To further predict behaviors in the future, we estimate a discrete time to collision $\hat{\tau}$ using their trajectories. This results in $\hat{\boldsymbol{x}}_{i,j} = \boldsymbol{x}_{i,j}^n + \hat{\tau}\boldsymbol{v}_{i,j}$. When further advanced in time by Δt, particles collide at $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$. (b) Projecting these collision constraints resolves the collision between $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$. (c) We compute the relative displacement d from time $\hat{\tau}$ to $\tilde{\tau}$. (d) d is decomposed into contact normal ($\mathrm{d}_{\boldsymbol{n}}$) and tangential ($\mathrm{d}_{\boldsymbol{t}}$) components. (e) The tangential contribution of the relative displacement is distributed to $\boldsymbol{x}_i^*$ and $\boldsymbol{x}_j^*$, which results in an avoidance resolution of future contacts.

where

$$a = \frac{1}{\Delta t^2}\|\boldsymbol{x}_i^* - \boldsymbol{x}_j^*\|^2, \tag{4.5}$$

$$b = -\frac{1}{\Delta t}(\boldsymbol{x}_i - \boldsymbol{x}_j)\cdot(\boldsymbol{x}_i^* - \boldsymbol{x}_j^*), \tag{4.6}$$

$$c = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 - (r_i + r_j)^2. \tag{4.7}$$

No potential energies associated with forces are required in our framework. To facilitate collision-free states in the future, we directly apply a collision-free constraint on future positions. Recall that in our simulation loop, the predicted position of particles i and j in the next time step are

$$\boldsymbol{x}_{i,j}^* = \boldsymbol{x}_{i,j}^n + \Delta t\boldsymbol{v}_{i,j}, \tag{4.8}$$

where $\boldsymbol{v}_{i,j}^b$ is defined in (4.1) , and we use the i,j subscripts to denote that the above is defined exclusively in the context of i or j.

We estimate a future collision state between i and j using τ. We first compute the exact time to collision using (4.4). Valid cases are those with $\tau > 0$ and $\tau < \tau_0$, where τ_0 is a fixed constant. We used $\tau_0 = 20$ in all our experiments. After pruning out invalid cases, we process the remaining colliding pairs in parallel (Section 6.1.1). We define $\hat{\tau} = \Delta t * \lfloor \tau/\Delta t \rfloor$,

where $\lfloor \cdot \rfloor$ denotes the floor operator. This is simply clamping τ to find a discrete time spot slightly before the predicted contact. With $\hat{\tau}$, we have

$$\hat{\boldsymbol{x}}_{i,j} = \boldsymbol{x}^n_{i,j} + \hat{\tau}\boldsymbol{v}_{i,j}. \tag{4.9}$$

Note $\hat{\boldsymbol{x}}_{i,j}$ are similar to $\boldsymbol{x}^n_{i,j}$ in the traditional collision constraint case (4.2) and are still in a collision free state. Stepping forward will cause the actual penetration. We define the colliding positions with

$$\tilde{\boldsymbol{x}_{i,j}} = \boldsymbol{x}^n_{i,j} + \tilde{\tau}\boldsymbol{v}_{i,j}, \tag{4.10}$$

where $\tilde{\tau} = \Delta t + \hat{\tau}$. We enforce a collision free constraint on $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$. Note that $\tilde{\boldsymbol{x}}_{i,j}$ is a function of $\boldsymbol{x}^*_{i,j}$; therefore, it is still essentially a constraint on $\boldsymbol{x}^*_{i,j}$. Due to its anticipatory nature, high stiffness on this constraint is not necessary. To prevent over-stiff behaviors, instead of using the overlap between the predicted particle locations, we define the stiffness to be $k\exp(-\hat{\tau}^2/\tau_0)$, where k is a user-specified constant.

4.2.5 Avoidance Model

We further present a novel avoidance model for crowd collision. The long-range collision constraint from Section 4.2.4 will cause agents to slow down due to motion along the contact normal from the collision resolve, which is often not desirable in dense scenarios (Figure 1.6). However, we observe that the tangential component of that collision response is often desired, effectively causing the agents to simply slide in response to the predicted collision. Hence, we preserve only the tangential movement in such collisions. We calculate the total relative displacement as

$$\mathrm{d} = (\tilde{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_i) - (\tilde{\boldsymbol{x}}_j - \hat{\boldsymbol{x}}_j), \tag{4.11}$$

which can be decomposed into contact normal and tangential components as follows:

$$\mathrm{d}_{\boldsymbol{n}} = (\mathrm{d} \cdot \boldsymbol{n})\boldsymbol{n}, \tag{4.12}$$

$$\mathrm{d}_{\mathbf{t}} = \mathrm{d} - \mathrm{d}_{\boldsymbol{n}}, \tag{4.13}$$

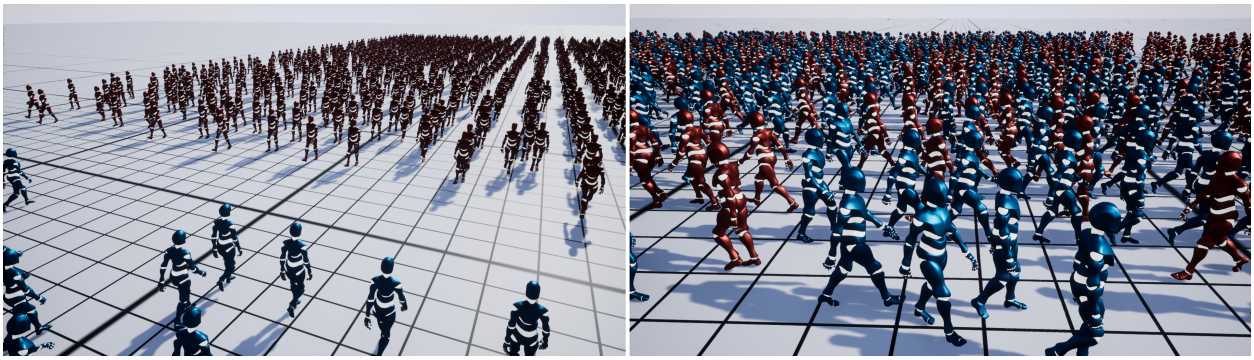


Figure 4.2: Two groups of agents passing through each other using our long range collision avoidance.

where $\boldsymbol{n} = (\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j)/\|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|$ is the contact normal. To this end, we preserve only the tangential component in the positional correction to $\boldsymbol{x}^*_{i,j}$. This provides an avoidance behavior and prevents agents from being pushed back in a dense flow. Figure 4.1 illustrates this process.

4.2.6 Maximum Speed and Acceleration Limiting

After the constraint solve, we further clamp the maximum speed and acceleration of the agents to better approximate real human capabilities.

4.2.7 Walls and Obstacles

Agents can interact with walls and other static obstacles in the environment. We prevent agents locomoting into walls and other static obstacles by a traditional collision response (4.2), between the agent's predicted position and the nearest point on the obstacle. The obstacle's collision point is assigned infinite mass.

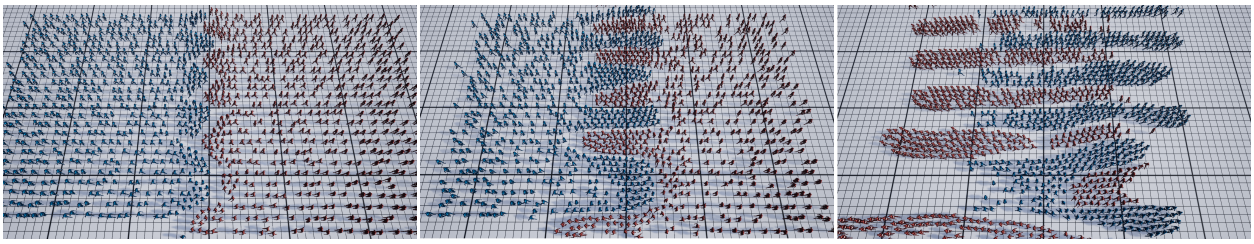


Figure 4.3: Groups passing each other using the avoidance model. Left: Groups of agents organize into a boundary front in preparation for collision avoidance. Middle: Agents huddle together in noticeable thick lanes. Right: Agents successfully pass each other.

CHAPTER 5

Layout Synthesis

The arrangement of objects into a layout can be challenging for non-experts, hence the existence of interior design professionals. Recent 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. In this chapter, we develop an alternative physics-based, continuous layout synthesis technique, which results in a significant gain in speed and is readily scalable.

Section 5.1 presents an overview of our synthesis approach, including the energy we use to track the quality of the layout. Section 5.2 presents the details and setup for our synthesis, including how to describe layout objects in a position-based framework, and the synthesis algorithm. We also formulate the different layout constraints used, their derivatives, and discuss their rationale in interior design.

5.1 Algorithm Overview

Our method explores the set of different layout arrangements by iteratively solving an input set of layout constraints. Starting from a uniformly distributed random initial position for each layout item, our method solves each constraint independently. Each constraint is associated with a spring-like stiffness, which determines the magnitude of the positional correction for satisfying the constraint. The positional corrections are either processed sequentially, or averaged in a batch (details in Section 5.2.3). To track the quality of the layout, we incorporate a Multidimensional scaling (MDS) (Cox and Cox, 2000) stress-like energy function:

$$E = \left(\sum_j w_j \left\| C_j(p_1, \ldots, p_n, \theta_1, \ldots, \theta_n) \right\|^2 \right)^{\frac{1}{2}}, \tag{5.1}$$

where C_j denotes the constraint, w_j is its corresponding weight, p_i denotes the layout object's position, and θ_i denotes the object's orientation.

5.2 Details of the Algorithm

5.2.1 Algorithm 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 three attributes:

- a position p_i,
- an orientation θ_i, and
- 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 type, either *unilateral* or *bilateral*.

Constraints are restrictions on the layout item's positions and orientations in the form of either equality of inequality. Equality and inequality are referred to as bilateral and unilateral constraints, respectively. A bilateral constraint is defined by the function

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

where the arguments denote the particle positions and orientations involved in the constraint.

Algorithm 3 Constraint Solver

```
1: for Object i do
2:     Initialize p_i = p_i^0
3:     Initialize θ_i = θ_i^0
4: end for
5: while solverIteration j < times do
6:     projectConstraints(C_1, ..., C_N)
7:     for Object i do
8:         generateCollisionConstraints()
9:     end for
10:    projectCollisionConstraints()
11: end while
```

By contrast, a unilateral constraint is defined by a function

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

Both types of constraints depend on either the position, or the orientation, or both, of the layout objects.

5.2.2 Layout Synthesis

Algorithm 3 overviews our method. Lines 2–3 initialize object i to a random location p_i^0 and orientation θ_i^0. In each iteration of the main loop, starting at Line 5, each constraint C_i is calculated and immediately projected, so that the next constraint uses the updated result (details in Section 5.2.5). The constraints $C_1, \ldots, C_N$ (details in Section 5.2.5.8) are fixed. Collision constraints require special treatment, since they may change in each iteration, and are generated in Line 8 using a spatial hash (details in Section 5.2.5.8).

Each constraint C_i has an associated stiffness parameter k_i. In each iteration, the stiffness either decreases, increases or remains constant, depending on the constraint type. For example, the pairwise distance constraint decreases over time, the collision constraint increases, and for hard constraints, like the layout boundary, the stiffness is constant. This is a type of numerical continuation method (Algower and Georg, 2003). Increasing the number of iterations results in more physically-plausible solutions. In addition, since some constraints

Order	1	2	3	4	5	6
Wall	-	-	●	-	-	●
Others	-	●	-	●	-	●
Collision	●	●	●	●	●	●

Table 5.1: Constraint projection order and frequency. In each iteration, a constraint is enforced depending on the above. Furthermore, our method resolves constraints partially, using an elasticity-inspired stiffness associated with each constraint. This combination allows the synthesized layout layout to satisfy conflicting constraints.

are conflicting, we evaluate them at different orders, by interleaving them, similar to Stam's proposal (Stam, 2009). Table 5.1 depicts the order we used for all scenarios.

5.2.3 Constraint Projection

Position-Based Dynamics (PBD) satisfies a system of non-linear positional constraints by solving each constraint independently. Given a constraint C and a particle i which is involved in the constraint, we derive a positional correction Δp_i is applied to particle i. According to Müller et al. (2007), this correction is derived by approximating each constraint function using

$$C(p_i + \Delta p_i) \approx C(p_i) + \nabla C(p_i) \cdot \Delta p_i. \tag{5.2}$$

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, the new position immediately visible to other constraints, or it is solved in a batch with other constraints, which we then average and project. 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. (2014b).

For constraint C, the positional correction for particle i is

$$\Delta p_i = -skw_i \nabla_{p_i} C(p_1, \ldots, p_N), \tag{5.3}$$

with scaling factor

$$s = \frac{C(p_1, \ldots, p_N)}{\sum_i w_i \left\| \nabla_{p_i} C(p_1, .., p_N) \right\|^2}, \tag{5.4}$$

and stiffness parameter k that is adjusted at each iteration. The stiffness of each constraint determines the importance of that constraint during a solve; i.e., a lower stiffness leads to less positional correction. We modified the original stiffness formula suggested by Müller et al. (2007) 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.

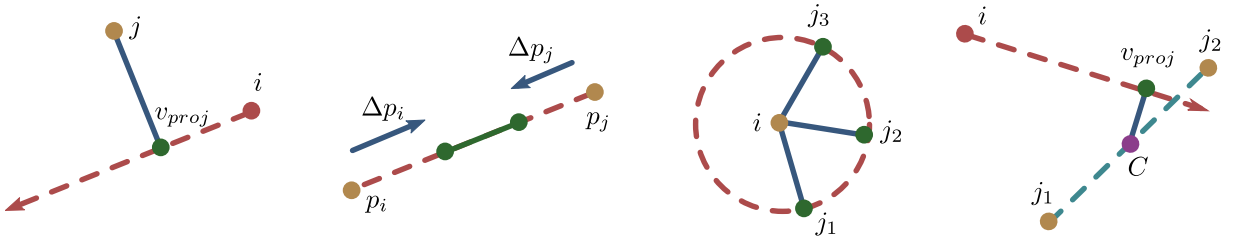
The orientation constraints are treated as a separate case. For these type of constraints, we find the shortest rotational correction to satisfy the constraint, and then apply this correction to the corresponding layout object's orientation. See Section 5.2.5.10 and Section 5.2.5.11 for more details.

5.2.4 Constraint Setup

Our method focuses on synthesizing layouts that conform to furniture arrangement constraints. We note that deciding the type and number of pieces of furniture to place is an orthogonal component to our constraint-based scheme. A user can select the type and number of objects for the layout and assign them to groups. Each group can be further assigned aesthetic or functional layout constraints, and a fixed zone of the layout space. Groups are reusable for defining other layout groups, and are also easily modifiable both in terms of number of participating layout objects, and in editing the group's layout constraints. Hard constraints such as observing layout boundaries and preventing collisions between layout objects are enabled by default. Complimentary to this selection step, the total number of objects and their relationship to their corresponding groups can also be sampled probabilistically from a predefined distribution; e.g., from realistic scene datasets (Silberman et al., 2012; Hua et al., 2016; Dai et al., 2017). This is similar to applying factor graphs (Yeh et al., 2012), but with a significant speed-up.

5.2.5 Layout Constraint Types

We formulate our constraint framework by adapting PBD to layouts and interior design standards (DeChiara et al., 2001). The following sections discuss the different constraints



(a) *Distance from vector* (b) *Distance* (c) *Focal point distance* (d) *Focal point symmetry*

Figure 5.1: Different positional layout constraints. Note that in (d), C denotes the center of mass of particles j_1 and j_2. v_{proj} denotes the projection of C onto the vector starting at focal point i.

employed, how they apply to layouts, and how they are projected.

5.2.5.1 Pairwise Distance

In interior design, two furniture items i and j (e.g., a chair and a table) are 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 the PBD framework to represent such a distance constraint between particles i and j that represent the two objects (Figure 5.1(b)):

$$C(p_i, p_j) = \|p_i - p_j\| - d, \tag{5.5}$$

where d is the desired distance between them. As explained by Müller et al. (2007), the positional corrections for particles p_i and p_j are

$$\Delta p_i = -\frac{w_i}{w_i + w_j}(\|p_i - p_j\| \ - d)\frac{p_i - p_j}{\|p_i - p_j\|} \tag{5.6}$$

and

$$\Delta p_j = \frac{w_j}{w_i + w_j}(\|p_i - p_j\| - d)\frac{p_i - p_j}{\|p_i - p_j\|}, \tag{5.7}$$

where w_i and w_j are the inverse masses of particles i and j.

5.2.5.2 Heat Point

For an object or a group of objects, the *heat point* is the position at which the group's center of mass is required to be. For example, a laptop should be located at the middle near the front side of a table for easy access. We define a bilateral constraint over all particles representing the objects using the center of mass concept:

$$C(p_1, p_2, \ldots, p_n) = \left\| \frac{\sum_{j=1}^{n} p_j m_j}{\sum_{j=1}^{n} m_j} - \hat{p} \right\|^2, \tag{5.8}$$

where $\hat{p}$ is the position of the heat point in the layout, and is manually defined by the user. When the corresponding parented layout surface moves, the heat point moves with the same correction.

5.2.5.3 Traffic Lanes

Objects should be arranged in a layout that accommodates traffic lanes (Jones and Allen, 2014; Deasy and Lasswell, 1990). Traffic lanes introduce space between objects to allow easy access to, e.g., different seating tiers. 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 position p_j of particle j and the vector starting from the position p_i of particle i and going in its orientation θ (Figure 5.1(a)):

$$C(p_i, p_j) = \left\| p_{v_{\text{proj}}} - p_j \right\| - d, \tag{5.9}$$

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 Figure 1.8. Note that if we do not want particle i to correct its position as a result of this constraint, we set $w_i = 0$.

5.2.5.4 Focal Point Symmetry

An object or a large piece of furniture can be deemed as a *focal point* for a group of items; e.g., the stage in a theater or a TV in a living-room (Jones and Allen, 2014; Deasy and Lasswell, 1990). We constrain a group of objects to be symmetric around a vector projected from the group's focal point—e.g., two chairs to be symmetric around the television's orientation—by calculating the center of mass of the surrounding particles j_k and projecting it onto the vector starting at the desired focal point and going in its orientation towards point $p_{v_{proj}}$. Then, we apply a bilateral constraint between $p_{v_{proj}}$ and the group's center of mass (Figure 5.1(d)):

$$C(p_i, p_{j_1}, \ldots, p_{j_n}) = \left\| \frac{\sum_{j=1}^{n} p_j m_j}{\sum_{j=1}^{n} m_j} - p_{v_{proj}} \right\|^2 . \tag{5.10}$$

The positional corrections follow Section 5.2.5.2.

5.2.5.5 Focal Point Distance

A group of objects might be constrained to be at a distance d from another object; e.g., a seating tier from a stage. We enforce such constraints by adding a bilateral distance constraint between the particle representing the focal point object i and the surrounding objects represented by particles $j_1, \ldots, j_n$ (Figure 5.1(c)). The constraint is implemented via the bilateral distance constraint (Section 5.2.5.1) in a pairwise fashion, for each pair $C(i, j_k)$, for $k = 1, \ldots, n$. It is also possible to prevent the focal point object from correcting its position in the manner shown in Section 5.2.5.3.

5.2.5.6 Layout Boundaries and Distance to Wall

Large furnishings usually work best when placed near a wall (Jones and Allen, 2014). For example, we usually do not want a bookshelf to be placed in the center of a room. Also, in a realistic use case, furnishings should not collide with the wall. Finally, all layout objects are constrained to the layout boundaries.

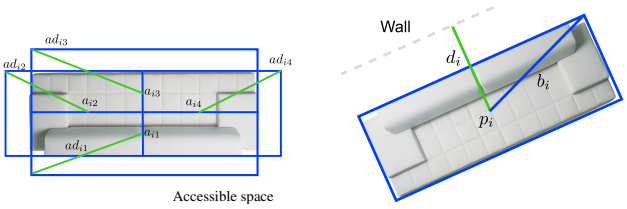


Figure 5.2: Left: ad_{ik} denotes the accessibility distance of the respective accessibility center a_{ik}. Right: Wall distance and orientation constraint. p_i denotes the center of object i and b_i denotes the bounding box radius.

Since the wall cannot be represented by a particle that can change its location, the wall acts like an external force on the constraint system. We define both unilateral and bilateral constraints for a object i constrained to be near a wall with distance d_i (Figure 5.2):

$$C(p_i) = \|p_i - p_{\text{wall}}\| - d_i, \tag{5.11}$$

where p_{wall} is the nearest wall point, and p_i is the position of the particle representing object i. In case of multiple possible wall points, we break the tie by picking the first point found. The positional corrections follow 5.2.5.1; however, note that since p_{wall} corresponds to a particle with infinite mass whose position cannot be modified, all positional corrections are applied to particle i.

5.2.5.7 Accessibility

Clearance between furniture items is essential for human comfort (Jones and Allen, 2014). For instance, a coffee table should be close, but not too close, to a sofa in a living-room. We propose a modified accessibility constraint, adapted from (Yu et al., 2011). Every object j is associated with a bounding box, where the faces of the bounding box that are orthogonal to the ground plane are identified with accessibility centers a_{jk}, where $k \in \{1, 2, 3, 4\}$ corresponds to the back, left, front, and right faces, respectively.

Our accessibility constraint is defined as a unilateral distance constraint d between the

accessibility center k of object j to the center of object i (Figure 5.2):

$$C(p_i, p_j) = \|p_i - a_{jk}\| - d, \tag{5.12}$$

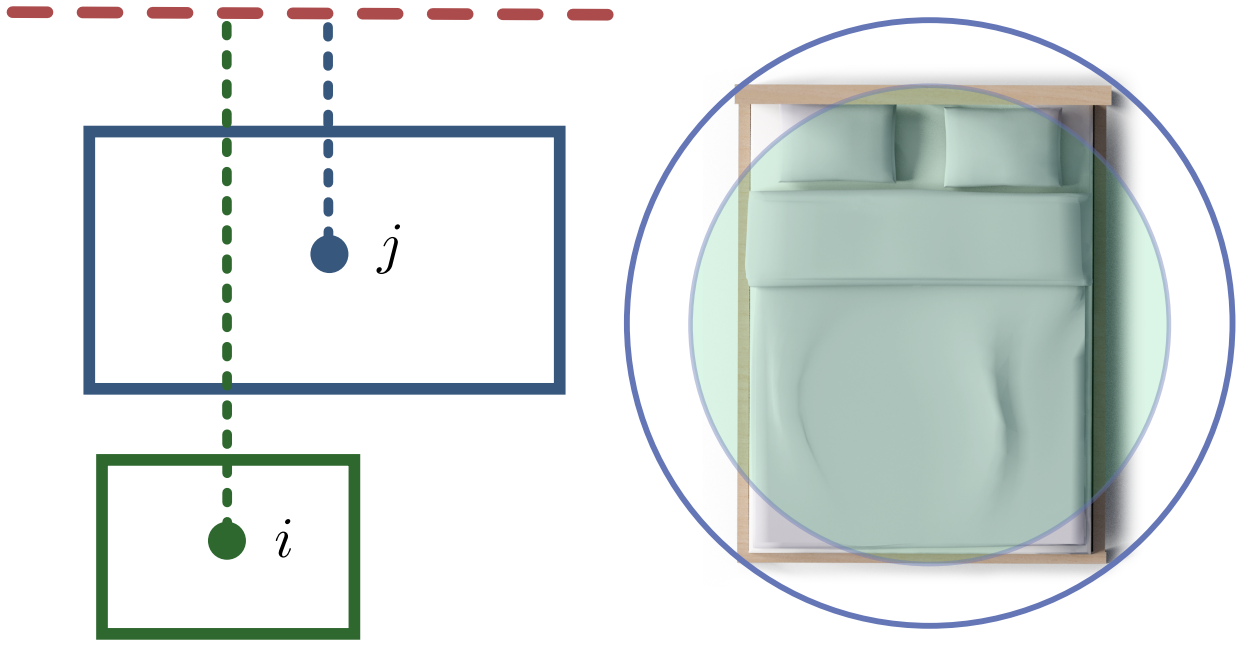
where p_i is the position of the particle representing object i, $d = b_i + ad_{jk}$, such that b_i is the diagonal of the bounding box for object i, and ad_{jk} is the diagonal of the respective accessibility center of object j. Note that our accessibility constraint is only activated if the respective cuboids defined by the accessibility distances intersect. The positional correction for particle j is applied as if accessibility point a_{jk} is rigidly attached.

5.2.5.8 Collision

The collision constraint is complementary to the accessibility constraint. Simply put, we want to enforce that the respective bounding cuboids of the layout objects do not collide, ensuring a realistic layout. To that end, we check for a collision between objects and, if so, we resolve the collision by approximation via a unilateral distance constraint between the bounding spheres (Figure 5.3b).

In a separate case, if during the constraint projection objects i and j are colliding, and both are constrained to be next to a wall (Section 5.2.5.6), we do an additional collision constraint projection between the respective two closest wall points w_i and w_j (Figure 5.3a) in order to satisfy both constraints, where w_i and w_j are *ghost* particles rigidly attached to the layout object's position, p_i and p_j. If there are multiple possible candidate wall points, we break the tie following Section 5.2.5.6.

The stiffness parameter of the accessibility constraint is constant. Since checking for all pairwise object collisions is computationally expensive, we employ a spatial hash (Teschner et al., 2003) in order to reduce the number of collision checks.



(a) (b)

Figure 5.3: (a) Additional collision constraint between two layout items i and j that are also constrained to be next to the wall. (b) Our method resolves collisions between layout item's by resolving collisions between the respective bounding spheres.

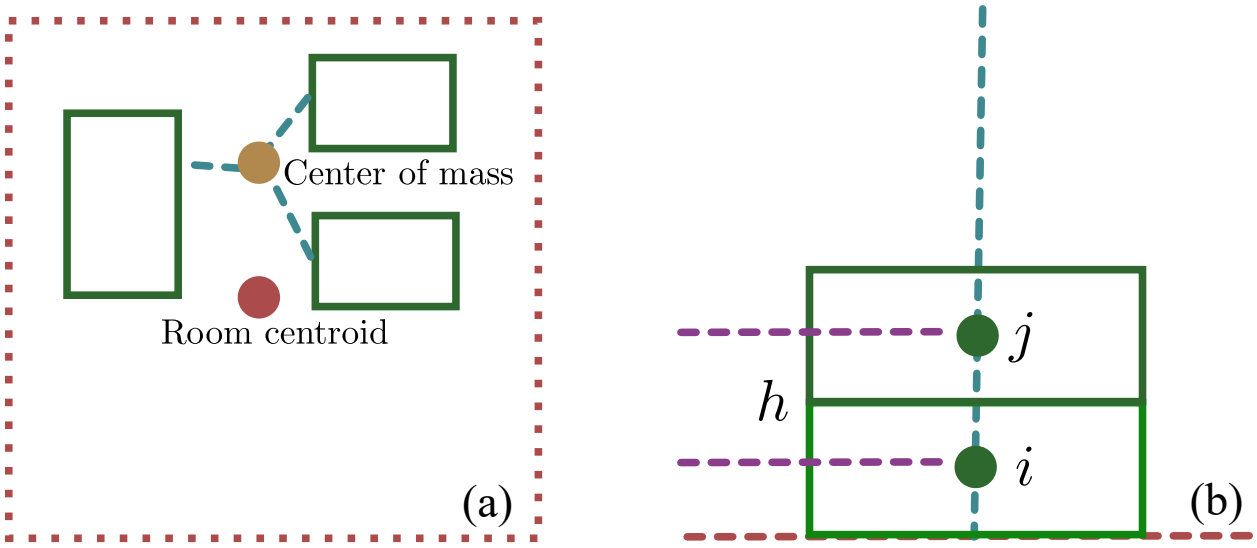
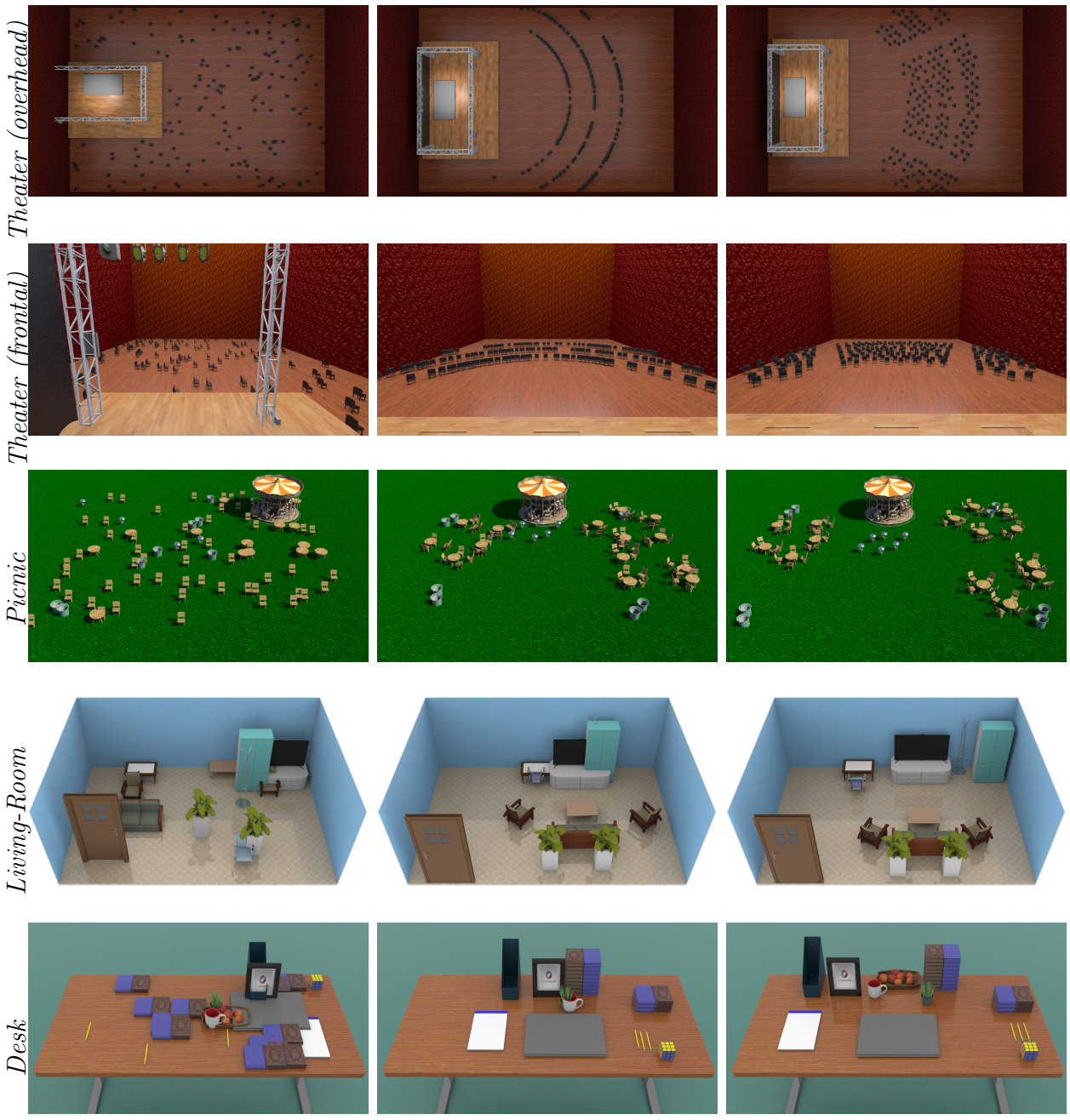


Figure 5.4: (a) Visual balance: layout object's center of mass and room centroid. (b) Stacking constraint: h denotes the vertical distance between the centers of objects i and j.



(a) Initial Layout (b) Intermediate State (c) Final Layout

Figure 5.5: Various synthesized layouts.

5.2.5.9 Visual Balance

To create a sense of equilibrium in a room, we want to arrange the furnishings such that the mean of the visual weights is close to the center of the room (Jones and Allen, 2014; Lok et al., 2004). For example, placing all the furnishings at one end of the room would create an imbalanced, inharmonious feeling. Since larger objects carry more visual weight, we define the visual weight of an object in accordance to its dimensions and simplify by evaluating the projection of the object's visual weight onto the ground plane.

Merrell et al. (2011) describe a visual balance constraint. Our method implements the visual balance as a bilateral constraint between the room's centroid and the center of mass of all the particles j, which represents the mean of the visual weights (Figure 5.4(a)):

$$C(p_1, p_2, \ldots, p_n) = \left\| \frac{\sum_{j=1}^{n} p_j m_j}{\sum_{j=1}^{n} m_j} - \hat{p} \right\|^2, \tag{5.13}$$

where $\hat{p}$ denotes the center of the room. Since the room's centroid is static, the particle corresponding to $\hat{p}$ is modeled to have infinite mass. The positional corrections are equivalent to those in Section 5.2.5.2.

5.2.5.10 Pairwise Orientation

In layout design, the orientation of an object to another object in the same furnishing group can induce more intimacy or improve functionality (Talbott et al., 1999). For example, a sofa should face a TV, a coffee table should be parallel to a sofa, and a seat in a theater should face the stage.

Yu et al. (2011) propose a pairwise orientation constraint between interacting layout objects. Similarly, We define a bilateral orientation constraint between interacting particles i and j. Let θ_i and θ_i' be the current and desired orientation of particle i towards particle j, and let θ_j and θ_j' be the current and desired orientation of particle j toward particle i. Then

the pairwise orientation constraint is defined by:

$$C_i(\theta_i, \theta_i') = |\theta_i - \theta_i'|; \quad C_j(\theta_j, \theta_j') = |\theta_j - \theta_j'|, \tag{5.14}$$

where we calculate the shortest rotational distance $\Delta\theta_i$ between the current and desired orientation for particle i, and similarly for particle j. This rotational correction is then applied to the particles with the corresponding stiffness k. For particle i, the corrected orientation is $\theta_i + k\Delta\theta_i$, and $\theta_j + k\Delta\theta_j$ for particle j. Note that the above orientation constraint does not change the particle positions p_i and p_j.

5.2.5.11 Orientation to Wall

Furnishings aligned to a wall leave the center of the room empty. Furthermore, some furnishings work best when placed parallel to the wall (e.g., a table or TV shelf (Jones and Allen, 2014)). Following the proposal of Yu et al. (2011), we formulate the orientation to wall constraint as a bilateral constraint between particle i to the nearest wall:

$$C(\theta_i) = |\theta_i - \theta_{\text{wall}}|, \tag{5.15}$$

where θ_i is the object's orientation in respect to the closest wall point, and θ_{wall} is the desired orientation to the wall. To satisfy this constraint, we calculate the direction and the shortest rotational difference between the desired θ_{wall} and the object's current orientation θ_i, and project it with the respective stiffness parameter, as in Section 5.2.5.10. In case of multiple possible wall points, we break the tie following Section 5.2.5.6.

5.2.5.12 Vertical Stacking

In layout design, accessories serve either a functional or decorative purpose (Jones and Allen, 2014). Vertical stacking is one common way to arrange accessories. For example, books may be stacked in order to conserve space.

A user can predefine the objects to be stacked. If object j is to be stacked on object

i, the vertical distance between the particles representing these objects should be equal to half the sum of the respective objects' vertical lengths $h = (h_i + h_j)/2$. We formulate the constraint as

$$C(p_i, p_j) = p_{j_z} - (p_{i_z} + h), \tag{5.16}$$

where z refers to the normal direction of the ground place. Hence, particle j should be placed above particle i (Figure 5.4(b)). Additionally, we constrain the other axial coordinates of particle j to be equal to those of particle i:

$$C(p_i, p_j) = |p_{j_y} - p_{i_y}|; \quad C(p_i, p_j) = |p_{j_x} - p_{i_x}|, \tag{5.17}$$

where x and y refer to the directions of the axes of the ground plane.

5.2.6 Differentiation of the Layout Constraints

Below we list the differentiation formulas we use in our method for the layout constraints developed in Section 5.2.5. For the sake of clarity, we repeat some of the constraint definitions.

5.2.6.1 Pairwise Distance

Let p_i, p_j be the positions of particles i, j, and d is the user-defined distance to be preserved. Then the derivatives for the pairwise distance constraint are:

$$\nabla_{p_i} C = \frac{p_i - p_j}{\|p_i - p_j\|}, \tag{5.18}$$

$$\nabla_{p_j} C = -\nabla_{p_i} C. \tag{5.19}$$

5.2.6.2 Heat point

Let $p_1, p_2, \ldots, p_n$ be the positions of the particles participating in the constraint, and let m_i be the corresponding masses. Then, the heat point constraint is:

$$C(p_1, p_2, \ldots, p_n) = \left\| \frac{\sum_{j=1}^{n} p_j m_j}{\sum_{j=1}^{n} m_j} - \hat{p} \right\|^2, \tag{5.20}$$

where $\hat{p}$ is the user-defined heat point. The differentiation for particle i is:

$$\nabla_{p_i} C = \frac{m_i}{\sum_{j=1}^{n} m_j} l, \tag{5.21}$$

where $l = (1, 1)^T$.

5.2.6.3 Traffic Lanes

Let p_i, p_j be the positions of particles i, j. We calculate $p_{v_{\text{proj}}}$ as the closest point on the unit vector v starting from the position p_i of particle i and going in its orientation: where $\mathbf{p}_{ij} = p_j - p_i$. We define $p_{v_{\text{proj}}}$ as a *ghost* particle that is the rigid extension of p_i. Therefore, any positional correction that affects $p_{v_{\text{proj}}}$ affects p_i, and $p_{v_{\text{proj}}}$ has the same mass has particle i. The differentiation is:

$$\nabla_{p_i} C = \frac{p_{v_{\text{proj}}} - p_i}{\left\| p_{v_{\text{proj}}} - p_i \right\|}, \tag{5.22}$$

$$\nabla_{p_j} C = -\nabla_{p_i} C, \tag{5.23}$$

which is similar to the pairwise distance constraint.

5.2.6.4 Focal Point Symmetry

Let particle i represent the focal point object, and let $p_{j_1}, \ldots, p_{j_n}$ be positions of the particles constrained to be around the focal point, so that the center of mass of these particles is: $\sum_{k=1}^{n} p_j m_{j_k} / \sum_{k=1}^{n} m_{j_k}$.

Following 5.2.6.3, We calculate $p_{v_{\text{proj}}}$ to be the closest point between the unit vector v starting from the position p_i of particle i and going in its orientation.

The constraint differentiation follows 5.2.6.2 for particle j_k:

$$\nabla_{p_{j_k}} C = \frac{m_{j_k}}{\sum_{k=1}^{n} m_{j_k}} l, \tag{5.24}$$

where $l = (1, 1)^T$.

5.2.6.5 Focal Point Distance

Similar to the Pairwise Distance constraint.

5.2.6.6 Layout Boundaries and Distance to Wall

Let p_i be the position of particle i, and let p_{wall} be the closest wall point. Then the differentiation constraint follows the pairwise distance constraint (replace p_j with p_{wall}). Note that since the wall is static, we assign infinite mass to p_{wall}. In case of multiple possible wall points, we break the tie by picking the first point found.

5.2.6.7 Collision

Let p_i, p_j be the positions of particles i, j that represent their corresponding layout objects, and let r_i and r_j be the respective bounding spheres (Figure 5.3b). The unilateral collision constraint is:

$$C(p_i, p_j) = \|p_i - p_j\| - (r_i + r_j). \tag{5.25}$$

The differentiation follows 5.2.6.1.

5.2.6.8 Accessibility

Let p_i, p_j be the positions of particles i, j that represent their corresponding layout objects, and let a_{j_k} be the accessibility center k of object j. We enforce an accessibility distance constraint between particles i and j:

$$C(p_i, p_j) = \|p_i - a_{jk}\| - d, \tag{5.26}$$

where $d = b_i + ad_{jk}$, such that b_i is the diagonal of the bounding box for object i, and ad_{jk} is the diagonal of the respective accessibility center of object j. We define point a_{jk} as a ghost point that is rigidly attached to p_j, so that any positional correction that affects a_{jk}, affects p_j. The constraint differentiation is:

$$\nabla_{p_i} C = \frac{p_i - a_{jk}}{\|p_i - a_{jk}\|}, \tag{5.27}$$

$$\nabla_{p_j} C = -\nabla_{p_i} C. \tag{5.28}$$

5.2.6.9 Visual Balance

Let $\hat{p}$ be the center of the room, $p_1, p_2, \ldots, p_n$ be the positions of the participating particles and let m_i be the corresponding mass of particle i. Then, the constraint differentiation follows 5.2.6.2.

CHAPTER 6

Experiments and Results

Section 6.1 presents experiments and results with our crowd simulation method, demonstrating that it is suitable for use in interactive games. In short, we provide examples of our method across different crowds scenarios typically found in games and visual effects, together with baseline benchmarks used to evaluate crowd simulation methods, such as bottleneck and evacuation benchmarks. We also show real-time performance in all of these settings, for both sparse and dense crowds involving up to 100,000 agents.

Section 6.2 demonstrates our layout synthesis method on a diverse set of examples and shows that it achieves results similar to conventional layout synthesis based on a Markov chain Monte Carlo (McMC) state-search, but is faster by at least an order of magnitude and can handle layouts of unprecedented size. Compared to our method that employs local gradient information for quick collision resolve, McMC-driven layout synthesis methods do not incorporate gradient information, and cannot easily resolve collisions between layout objects in tight spaces. Results involving tight synthesis spaces are presented.

6.1 Crowd Simulation

6.1.1 Setup and Parameter Settings

We implemented our crowd simulation framework in CUDA, using an NVIDIA GeForce GT 750M. We set $\Delta t = 1/48$ sec for all the experiments (2 substeps per frame). We solve each constraint group in parallel, employing a Jacobi solver, with a delta averaging coefficient of 1.2. To find neighboring agents, we use two hash-grids, for short and long range collisions. This is more efficient than using one grid for both, since the long range grid covers a bigger

	# agents	LR	A	ms/frame
Sparse passing	1,600	On	-	11.27
Sparse passing	1,600	-	On	11.61
Dense, low count	1,600	On	-	12.03
Dense, low count	1,600	-	On	11.34
Dense, high count	10,032	On	-	14.06
Dense, high count	10,032	-	On	13.63
Bears and Rabbits	1,152	-	On	11.86
Dense Ellipsoid	1,920	-	On	10.06
Proximal Behavior	50	On	-	10.12
Proximal Behavior	50	-	On	10.13
Target Locomotion	192	On	-	10.42
Bottleneck	480	-	-	11.99
Bottleneck	3,600	-	-	17.76
Bottleneck	100,048	-	-	43.66

Table 6.1: Timings. LR: long range collision constraint; A: avoidance model constraint. All experiments use $\Delta t = 1/48$, with 6 iterations per time step. These timings do not include rendering times.

collision radius. Each grid is constructed efficiently and in parallel. See (Green, 2008; Macklin et al., 2014a) for additional details.

In our simulations, we use 1 stability iteration to resolve contact constraints possibly remaining from the previous time step, and 6 iterations in the constraint solve loop. Additional iterations can increase stability and smoothness, but at increased computational cost.

For agent rendering and locomotion synthesis, we used Unreal Engine 4.15. For smooth locomotion, we clamped the agent's skeletal positional acceleration and rotational velocity. Additionally, we applied a uniform motion scaling of about 30. We rendered the motion at about 5 times the simulation rate.

We demonstrated the robustness of our position-based framework in a variety of scenarios. To simplify the experiment setup and unless otherwise stated, we modeled all agents using a disk with radius 0.5, and use the same width for our humanoid agents in the rendering stage. For smoother motion, we allow an expansion of the agent's disk radius by 5% during collision checks. For each benchmark, we used a simple preferred velocity planner, where the preferred velocity of each agent points to the closest user-scripted goal. We also slightly varied the preferred velocity of each agent around a mean of 1.4, to achieve a more realistic simulation. Table 6.1 presents timing information.

6.1.2 Benchmarks and Analysis

6.1.2.1 Sparse Passing (Low Count, Long Range Collision)

We experimented with two groups of agents locomoting in opposite directions (Figure 4.2). The agents in each group are positioned in a loose grid formation with an initial separation distance. To avoid collisions, the agents use the constraint of Section 4.2.4. In this scenario, the agents organize into narrow lanes, and pass each other easily.

6.1.2.2 Sparse Passing (Low Count, Avoidance)

This scenario is identical to 6.1.2.1, but the agents employ the constraint of Section 4.2.5 to avoid collisions. In this scenario, the agents form thicker lanes (Figure 4.3), which accumulate into different groups.

6.1.2.3 Dense Passing (Low Count, Long Range Collision)

A total of 1,600 agents are split into two groups, with a separating distance of 2.5 (Figure 6.1). We used a higher and denser crowd of agents. To avoid collision, the agents employ the constraint of Section 4.2.5. Because of the dense agent setting, the two agent groups do not easily pass each other, and some bottleneck groups are formed. Eventually, the agents pass, avoiding unrealistic collisions.

6.1.2.4 Dense Passing (Low Count, Avoidance)

This experimental setup is identical to 6.1.2.3. To avoid collision, the agents employ the constraint of Section 4.2.5. In this scenario, the agents form thicker lanes, which form into different groups (Figure 6.1).

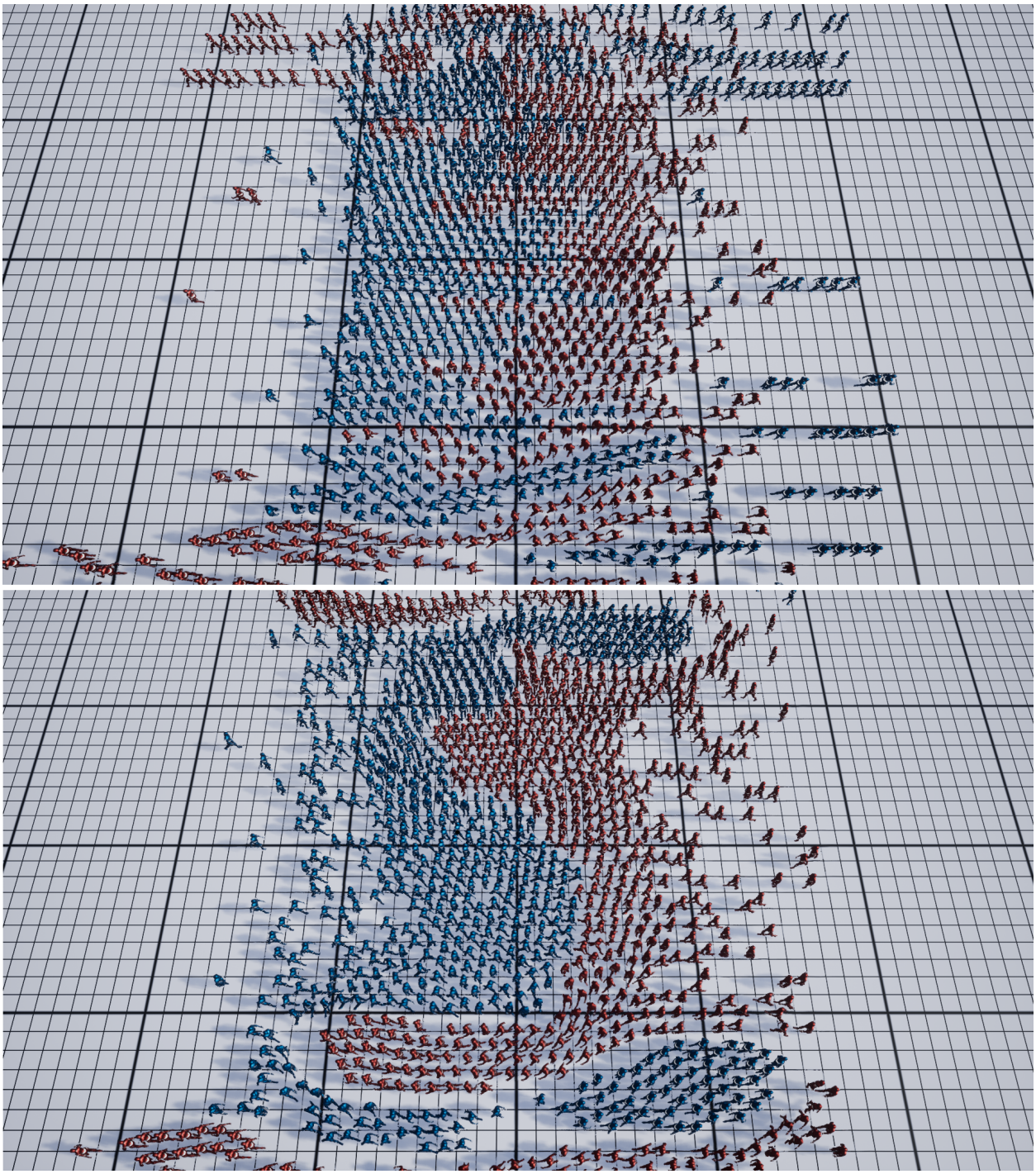


Figure 6.1: High density agent simulation. Top: Long range collision. Bottom: Avoidance model.

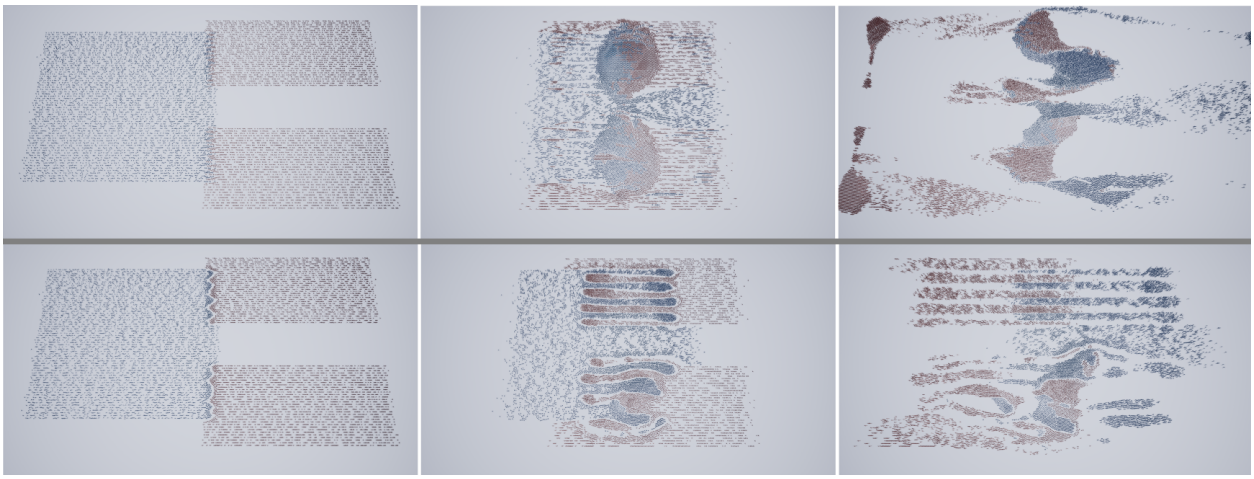


Figure 6.2: High density and high agent count. Top Row: Agent groups avoid each other using Long Range Collision. Bottom Row: Using the Avoidance model.

6.1.2.5 Dense Passing (High Count, Long Range Collision)

A total of 10,032 agents are split into two groups (Figure 6.2) with a separating distance of 3.5. This experiment setup is identical to 6.1.2.5.

6.1.2.6 Dense Passing (High Count, Avoidance)

This experiment setup is identical to 6.1.2.5. To avoid collision, the agents employ the constraint of Section 4.2.5. In this scenario, the agents form thicker lanes, which form into different groups.

6.1.2.7 Bears and Rabbits

In this experiment, we showcased how a Lagrangian PBD scheme may be employed to model agents of different sizes (Figure 6.3). We modeled a group of rabbits passing through a group of bears, totaling 1,152 agents. The rabbits had size 1.0, while the bears had a size ranging from 2.5 to 4.0. To simulate that bears are less prone to change their path than rabbits, we assigned the bears a mass that is approximately 30 times greater than that of the rabbits.

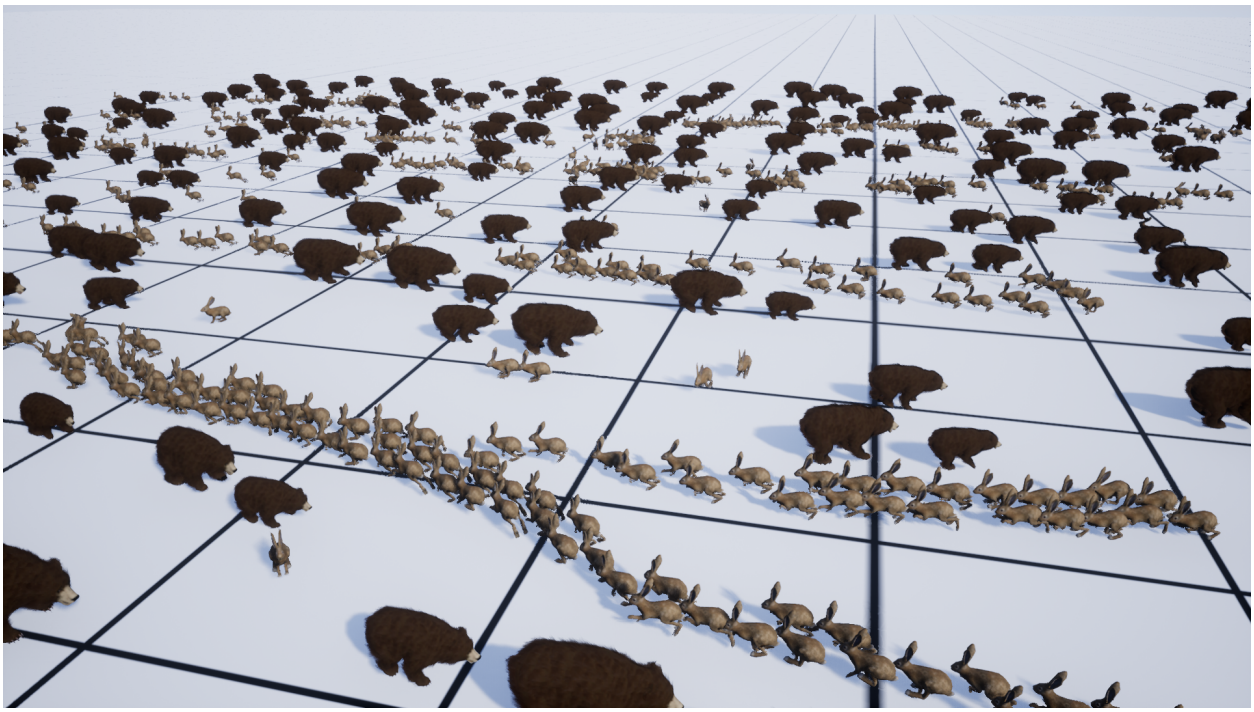


Figure 6.3: A group of smaller agents (rabbits) passing through a group of larger ones (bears).

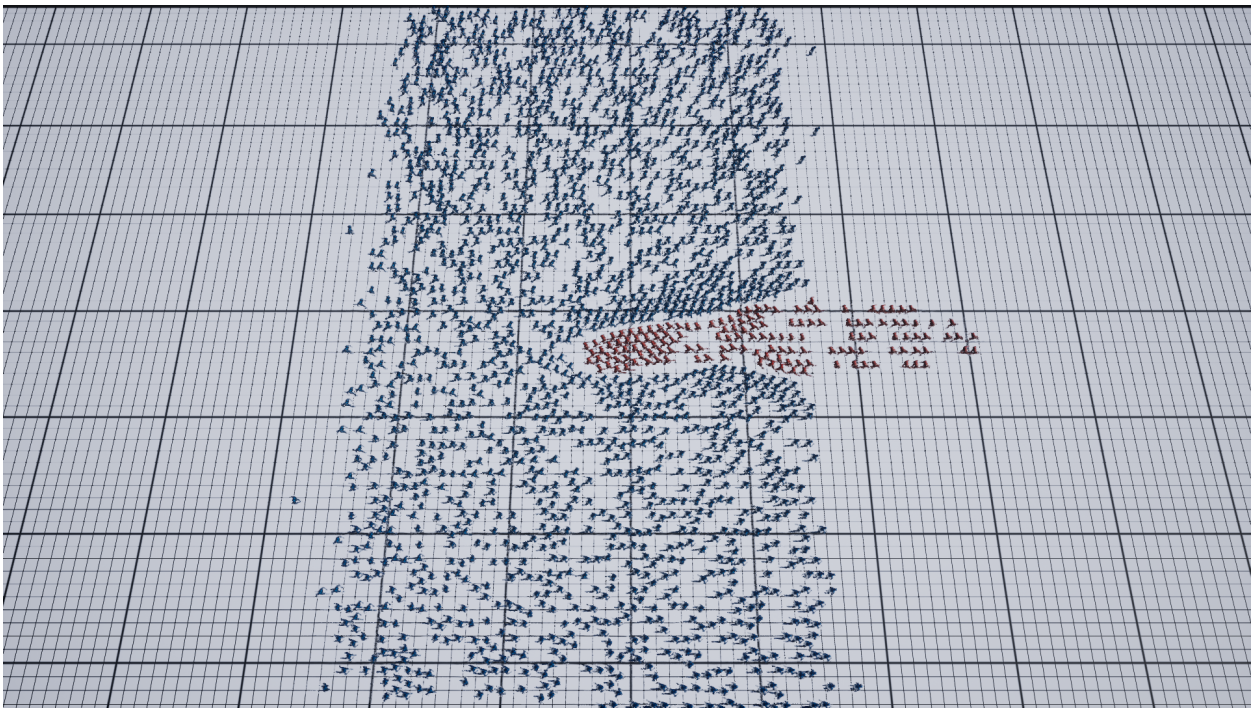


Figure 6.4: A small ellipsoid shaped group passing through a larger group.

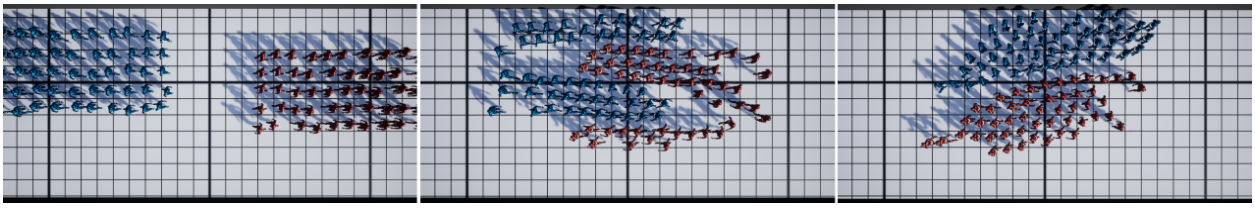


Figure 6.5: Proxemic group behavior. Left: Initial state. Center: Agents avoid each other using the long-range collision model, while creating lanes. Right: Agents avoid each other using the avoidance model.

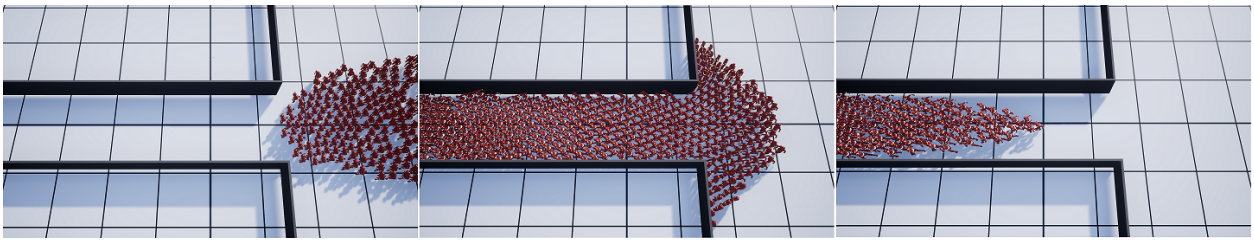


Figure 6.6: A group of agents passing through a narrow corridor. Left: Agents huddle on approaching corridor's entrance. Middle: A semi-circular arch forms as agents enter a narrow corridor. Right: Agents successfully exit.

6.1.2.8 Dense Ellipsoid

This simulation comprises 1,920 agents. To reach their goals, an ellipsoid-shaped group of agents (Figure 6.4), with an initial separation distance of 3.3, must locomote through a larger, rectangular group of agents, with a separation distance of 3.0. Throughout the entire simulation, the small group retains its shape and successfully passes the larger group.

6.1.2.9 Proximal Behavior, Avoidance Model

Two groups of 50 agents start in tightly packed formations, and must pass each other in a narrow hallway with limited collision avoidance space (Figure 6.5). This benchmark demonstrates that our novel avoidance model creates proxemic behavior in agent groups (He et al., 2016).

6.1.2.10 Proximal Behavior, Long Range Collision

Here, we used the same setting as 6.1.2.9. We observed lane formation and splitting of the original group.

6.1.2.11 Target Locomotion, Long Range Collision

192 agents start in a uniform random grid setting at a separation distance of 5.5. The locomotion targets are in a similar, but in a translated grid pattern, randomly perturbed with additive uniformly distributed random noise. The objective of this benchmark was to show that agents are able to reach their respective goal with minimal interference.

6.1.2.12 Bottleneck

We demonstrated our method on a bottleneck scenario with varying number of agents. Agents must pass through a narrow corridor to reach their goal (Figure 6.6). In this scenario, we observed jamming and arching near the corridor's entrance, as well as the formation of pockets, a phenomena observed in realistic crowds, which was also reported in (Golas et al., 2014; Guy et al., 2010a).

6.1.3 Comparison

The method described by Karamouzas et al. (2014) is considered the state-of-the-art model for explicit force-based modeling of pedestrian behavior, and it has been validated against human behavior. We implemented this method based on code obtained from the authors. For the comparison, we chose the same parameter settings and timestep as in our method (Section 6.1.1). Using 1,344 agents, we preformed experiments in the two following settings (Figure 6.7):

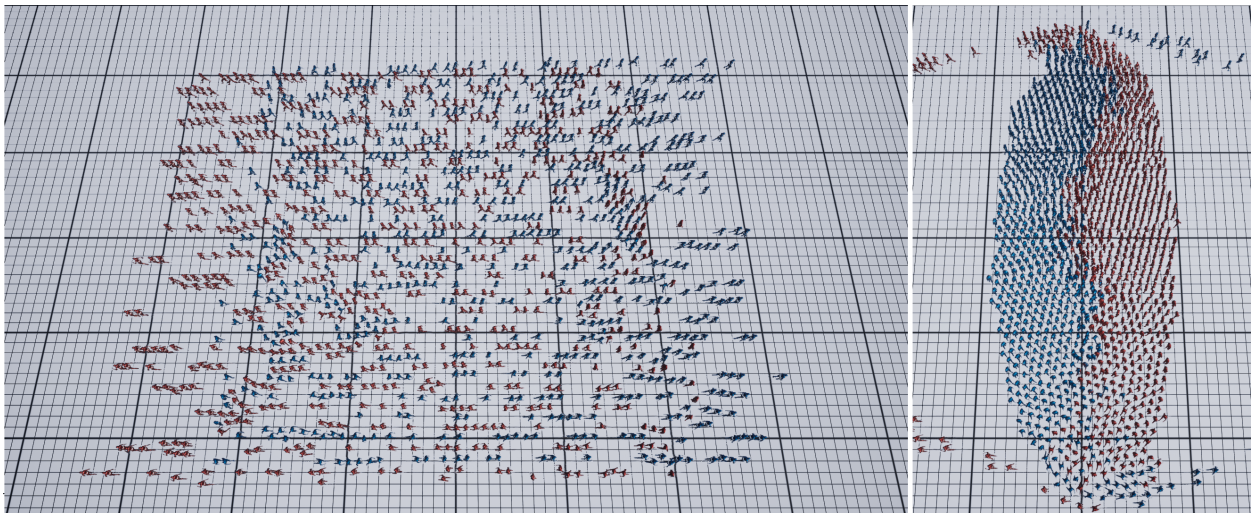


Figure 6.7: Explicit force-based power law (Karamouzas et al., 2014). Left: In a sparse setting, the agents successfully avoid collisions. Right: In a dense setting, the agents collide, overlap, and are not able to pass smoothly.

6.1.3.1 Crowd Passing (Sparse)

For the sparse setting, we used a separating distance of approximately 4.5 between agents. Agents preformed well and avoided collisions, managing to pass with minimal interference to the opposing group. Lane patterns emerged.

6.1.3.2 Crowd Passing (Dense)

In the dense setting, we used a separation distance of approximately 3.3. In this setting, the agents were not able to maintain their trajectory or avoid collisions with the opposing group. Some of these collisions were not resolved, leading to an unrealistic state for almost half of the simulation. Our supplemental video offers additional details.

6.1.4 Discussion

Our crowd simulation method demonstrated interesting group interactions, such as groups passing each other seamlessly, as well as the formation of traffic lanes and subgroups with minimal interference. We demonstrated our novel PBD method on groups of agents of various sizes, arranged in varying densities, using different mixtures of PBD constraints.

We presented novel long range collision constraints with adaptive stiffness, which serve as a realistic preconditioner for the actual collision from frictional contact, with a sufficient stiffness that enforces non penetration. Our solution is flexible and produces interesting patterns and emergent behavior. Compared to existing methods, the advantages of PBD are large time steps, guaranteed stability, and ease of control. In addition, our approach allows simple integration into a preexisting PBD framework. By adding new constraints, our robust, parallel framework can easily incorporate more complex crowd behaviors with minimal run time cost.

From the above experiments, we noticed that the power law method does not provide a collision-free model for dense crowds. Nevertheless, careful parameter tuning or increasingly small time steps may help, albeit at the expense of efficiency and ease of use.

6.2 Layout Synthesis

We used Python and Cython to implement our layout synthesis technique. We ran our experiments on a 2.5 GHz Intel i7 Macintosh system.

Our experimental scenarios are shown in Figure 5.5. For all our demos, we used a uniformly sampled random initial guess as shown in the first frames of the figures illustrating our examples. The initial object locations and orientations in each experiment were set randomly. This corresponds to Line 2–3 of the main loop in Algorithm 3. Accessibility and collision constraints apply and are generated in all experiments. We then ran our method with the constraints described below. As weights for the layout's energy function (5.1), we chose 20.0 for the wall constraints, 150.0 for collision and accessibility constraints, and 1.0 for all the rest, since the wall, collision and accessibility constraints are treated as hard constraint. We determined these weights experimentally, by the order of importance of the constraint type. The method terminates the iterative procedure when there has been no improvement to the minimum layout energy for the past 50 iterations. The experiments in Section 6.2.1 run for about 150–300 iterations.

Table 6.2 reports run-times of our experiments. The bottleneck of our algorithm is in

	# Objects	Our Method(sec)	McMC-SA(sec)
Theater	201	39.50	5852
Picnic	77	4.77	253
TP Picnic	53	2.42	109.43
Desk	21	0.69	37.06
TP Bedroom	12	0.67	22.31
Living-Room	10	0.62	26.78

Table 6.2: Comparing the run-time of our method versus baseline McMC-SA method. TP denotes Tightly-packed. The timings are the mean of 10 runs, where both our method and McMC have the same starting conditions.

solving the accessibility and collision constraints, as shown in Figure 6.8. The supplementary document provides the additional details of the constraints used in each of the above scenarios.

6.2.1 3D Layout Synthesis

In this section, we describe the experimental scenarios of Figure 5.5.

6.2.1.1 Theater

We demonstrated the efficacy of our method by running our algorithm in a theater scene with 200 seats and a stage. For symmetry, we set the initial location of the stage to be halfway across the theater's width. The seats are divided into groups, with each seat belonging to a different seating tier. We employed a focal point constraint between each seating tier and the stage. All seats were influenced by the traffic lane constraint for stage pathways. We further tested the scalability of our algorithm by running more experiments in which we modified the number of seats. Figure 6.9 plots the resulting run-times. Note that the chairs assigned to each seating tier do not have a specified distance constraint within the tier. Instead, the distances are enforced through the accessibility and collision constraints. Even without such distance constraint, the chairs are able to maintain an almost regular distance with other chairs in the tier.

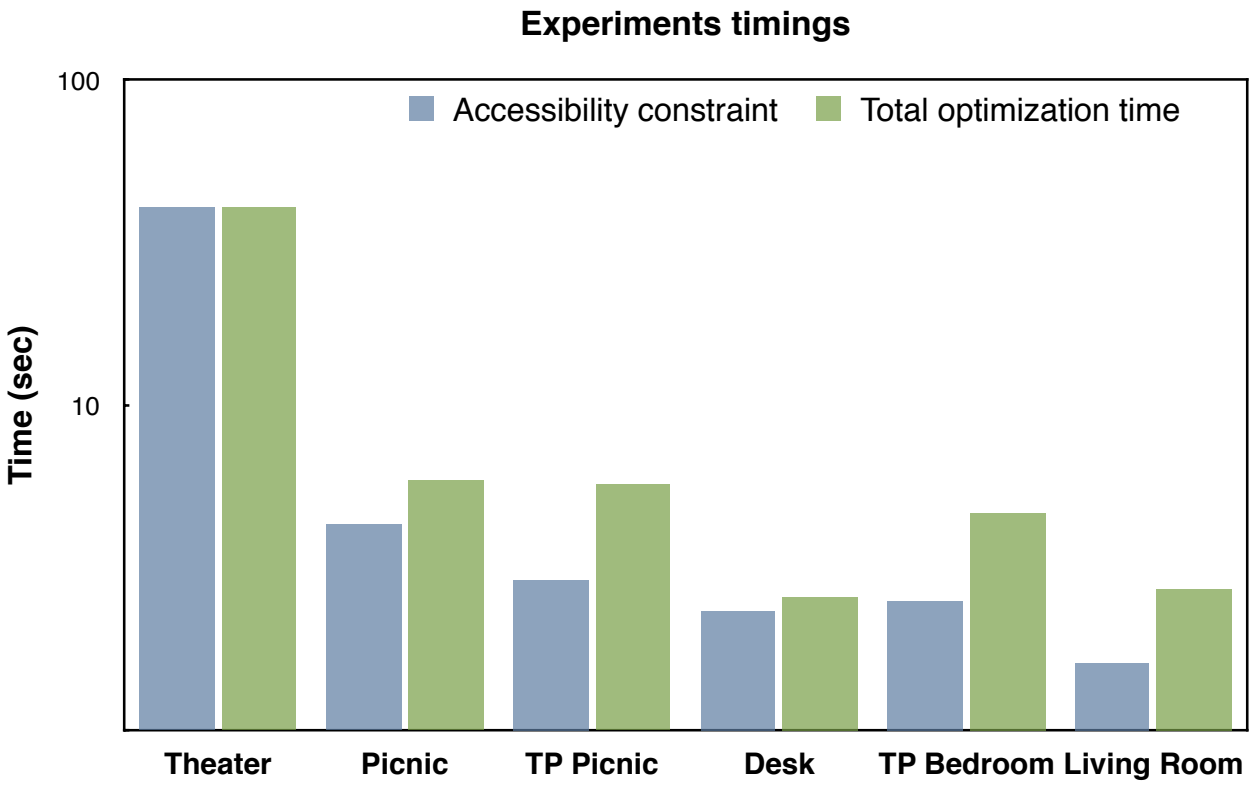


Figure 6.8: Run-times. TP denotes Tightly-packed. The major computational costs stems from resolving the accessibility and collision constraints, especially when increasing the number of layout objects.

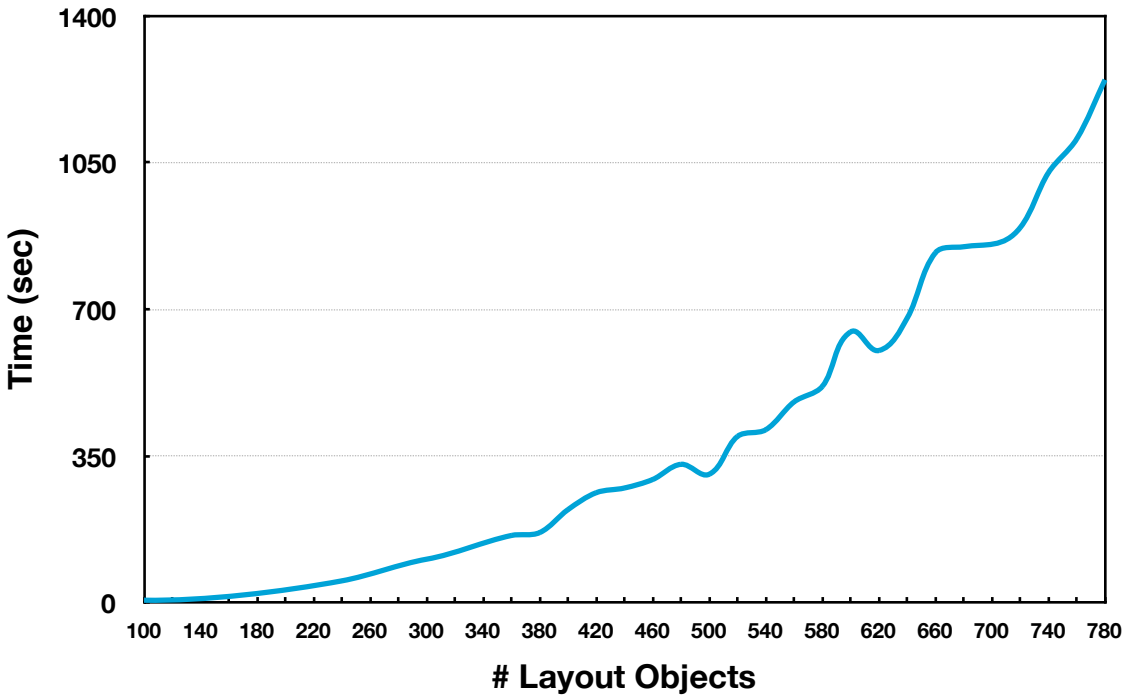


Figure 6.9: Run-time vs an increasing number of theater seats.

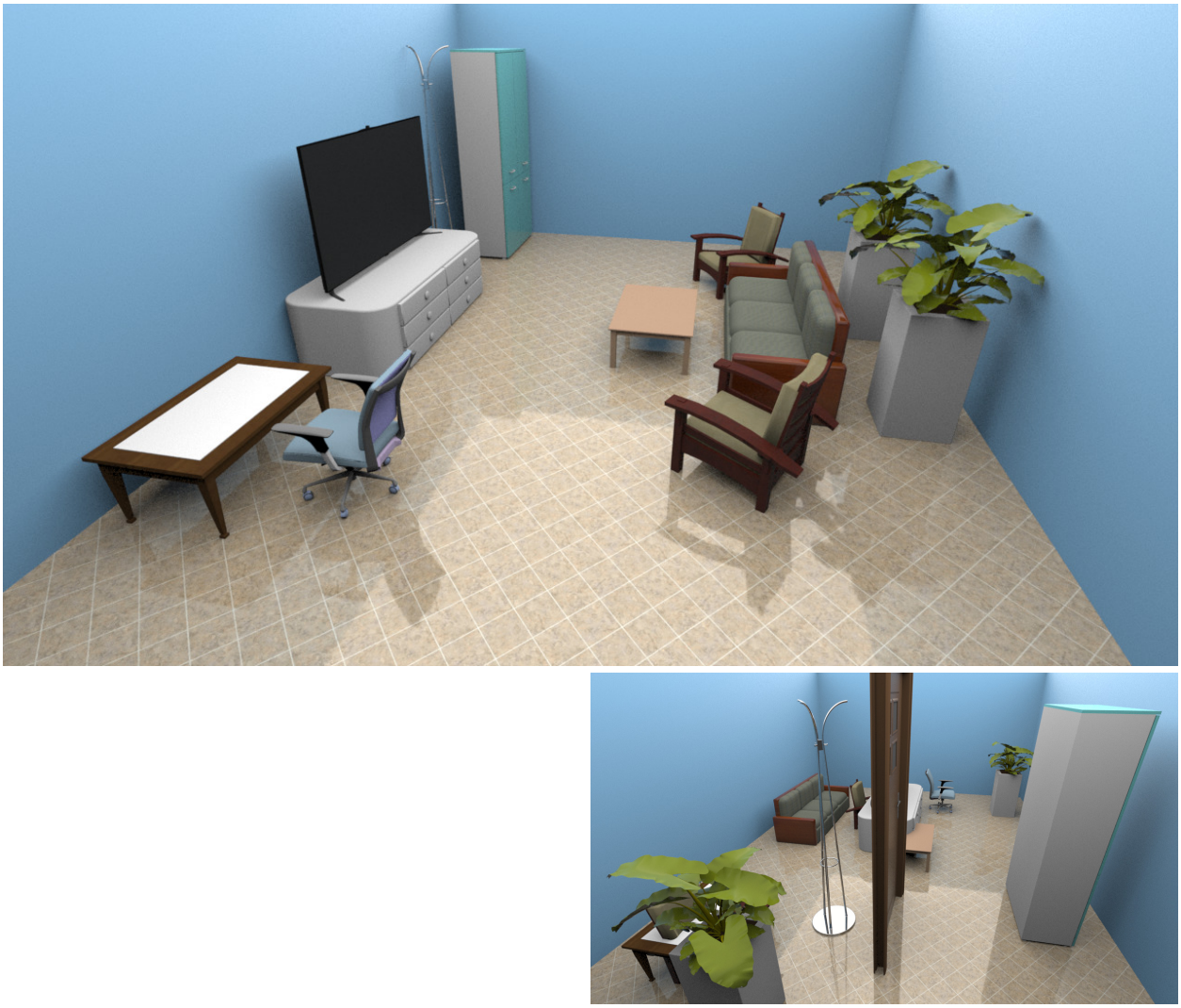


Figure 6.10: Optimized living room layout satisfying criteria such as distance, viewing angle, focal point grouping, and visual balance, starting from the initial random layout shown beneath.

6.2.1.2 Picnic

The picnic scene consists of 14 tables, 48 chairs, 8 trash cans, 6 BBQ grills, and a carousel. The main constraints are focal point constraint between each group of chairs and the respective table, distances between tables, distances between chairs around tables, and heat point constraint to determine the locations of BBQ grills, trash cans, and picnic tables.

6.2.1.3 Living-room

The living-room layout (Figure 6.10) contains 2 chairs, 2 indoor plants, a sofa, a coat rack, a door, an office desk, an office chair, and a TV. The main constraints are focal point constraints between the TV, sofa, and chairs, as well as wall constraints on the big furniture objects and plants.

6.2.1.4 Desk

We also demonstrated the performance of our algorithm on a tightly packed desk with small objects, including 12 books, 3 pencils, a food plate, a binder, a photo frame, a potted plant, a laptop computer, and a mug. The main constraints are focal point constraints between certain objects, heat point constraints on different desk parts, and a stacking constraint for the books.

6.2.2 Layout Synthesis in Tightly-Packed Scenarios

Our method can also cope with highly constrained and tightly-packed settings.

6.2.2.1 Tightly-packed bedroom

The tightly packed bedroom contains multiple beds and pieces of furniture. The beds, bookcase and table are constrained to be next to the wall. The coat rack is constrained to be at a certain distance from the bookcase. We demonstrate that using different initial conditions results in different suggested layouts. Even though the space is tight, our method successfully synthesizes different layout suggestions (Figure 6.11).

6.2.2.2 Tightly-Packed Picnic

This tightly-packed setting demonstrates our method's ability to synthesize diverse layouts with different numbers and types of furniture objects. We synthesize a tightly-packed picnic scenario in two stages. In the first stage, we randomly vary the number of layout objects of

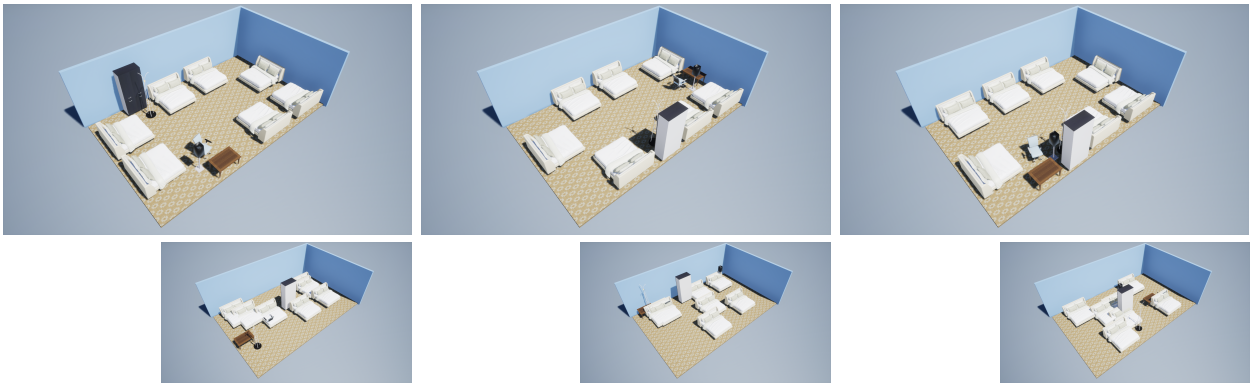
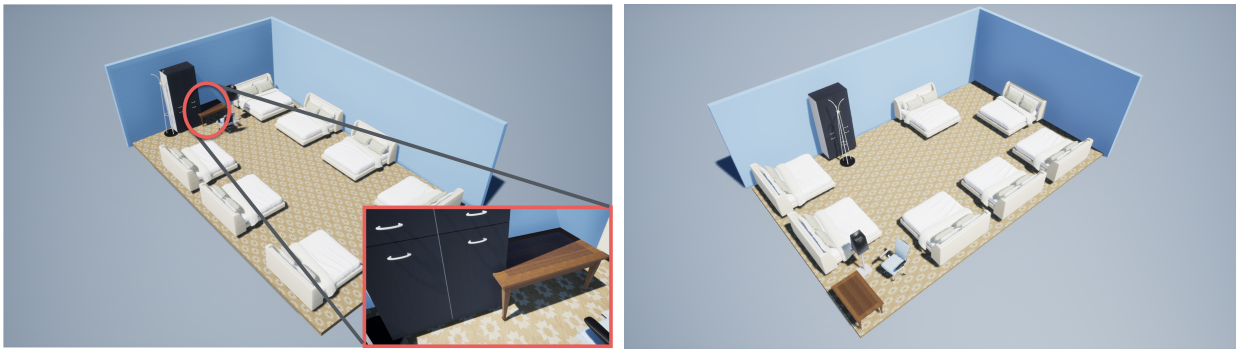


Figure 6.11: Our method produces different layout suggestions (top) by initializing from different random initial conditions (bottom).



(a) (b)

Figure 6.12: (a) A baseline McMC-SA approach struggles with tightly-packed bedroom. In McMC-SA, layout objects correct their position using probabilistically sampled shift moves. This results in configurations where objects are "locked" in configurations that are probabilistically hard to escape from, although a different initialization or more cleverly designed moves might help. (b) Our method does not.

each type, similar to Yeh et al. (2012). The available layout objects are a superset of the previous picnic scenario, with an additional rectangular picnic table. For a more uniform layout, we rigidly attached 4 chairs to each round picnic table. In the second stage of the synthesis, we run our method for 270 iterations. Figure 6.13 shows these synthesized layouts.

6.2.3 Application: Layout Synthesis from Real-World Images

We have applied our layout synthesis technique to bridge the gap between the physical and virtual worlds: Given an input image of an interior or exterior space, and a general user specification of the desired furnishings and layout constraints, we have implemented a

Figure 6.13: Diverse tightly-packed picnic layouts synthesized by our algorithm. Each synthesized layout has a different number and types of objects.

(a) (b)

Figure 6.14: (a) Image of a vacant living room. (b) Image augmented with a layout synthesized by our method.

system that can automatically furnish the scene with a realistic arrangement and display it to the user by augmenting the original image (Figure 6.14 and Figure 6.15). Our system can deal with varying layouts and target arrangements at interactive rates, which affords the user a sense of collaboration with the design program, enabling the rapid visual assessment of various layout designs, a process which would typically be time consuming if done manually. Our method is suitable for smartphones and other camera-enabled mobile devices.

Appendix C describes our system in detail.

6.2.4 Comparison

We compared the performance of our method to a baseline layout synthesis approach based on simulated annealing with a Metropolis-Hastings McMC state-search step, which we denote

(a) (b) (c)

Figure 6.15: Outdoor Yard layout. (a) Random initial state; chairs are colliding unrealistically. (b) Intermediate state of the optimization; not all chairs are facing their respective tables. (c) Final optimized layout; all chairs are in their correct positions and orientations.

McMC-SA. Our implementation is based on code obtained from the authors of (Yu et al., 2011). In this implementation, the proposal function shifts an attribute of one layout object in each state-search step. We employed the same energy function to track the quality of the synthesized layouts, and ran the comparison several times, with different conditions, such as different temperature changes to the annealing algorithm and differing constraint weights. For all experiments, we used a linear and evenly spaced annealing schedule, for about 20,000 iterations, with the additional stopping condition in case the energy function value did not improve by more than 0.1% during the last 1500 iterations.

Experimentally, we noticed that a baseline McMC-SA approach has trouble accommodating tightly-packed and constrained layouts, like in the tightly-packed bedroom and picnic scenarios described in the previous section. One of the possible explanations is that this approach does not directly use local constraint gradient information for shifts between layout configurations, but rather a global energy function which is limited in scope. Initially, the synthesized layout looks satisfactory. However, upon closer inspection, the layout had unresolved collisions (Figure 6.12). This is illustrated by Figure 6.16, which plots the energy functions of both methods for the Tightly-Packed Picnic scenario.

Nevertheless, McMC-SA has a theoretic probabilistic chance to escape these collisions by choosing different parameter settings, using more complex hand-crafted McMC-SA shifts moves, as well as tuning of the energy weights of different constraints. Unfortunately, neither

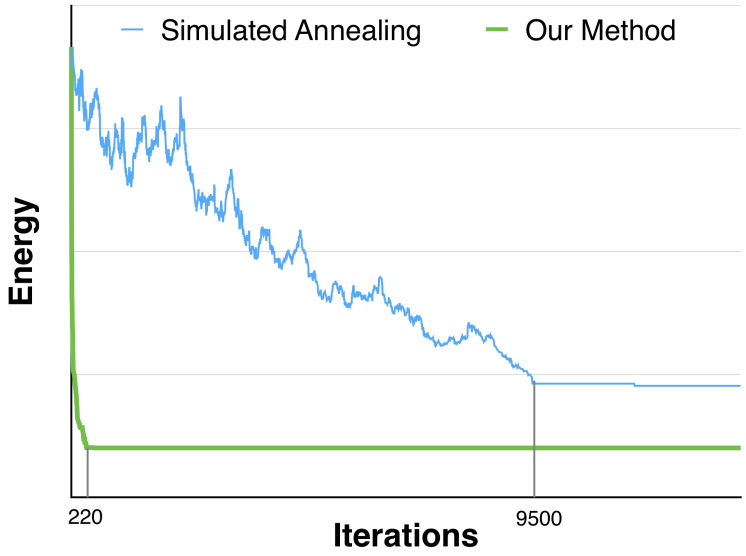


Figure 6.16: Energy plot (linear scale) for tightly packed picnic layout, running a McMC-SA implementation, compared to our method. Total run-time for McMC-SA is approximately 163 seconds, versus 4.7 seconds for our method. Within 220 iterations, our method immediately converges to a suitable layout, while in terms of energy, McMC-SA converges to a less satisfactory layout around 9500 iterations.

of the settings we experimented with provided a collision-free layout suggestion in a tightly-packed setting.

Computationally, we observed that McMC-SA is slower by at least an order of magnitude compared to our method (Table 6.2). The computational cost increases dramatically with the number of objects to be synthesized. Hence, our method is more suitable for interactive applications.

6.2.5 Discussion

This work is the first to demonstrate that certain principled, continuous simulation methods popular in physics-based animation can produce satisfactory layout synthesis results at extremely low computational cost. Layout synthesis objectives are represented by a set of positional constraints, each having an associated spring-like stiffness. Our method solves the conflicting set of constraints by exploiting the efficiency and speed of PBD. It yields fast layout synthesis with quality similar or better to that provided by previous probabilistic, McMC-based alternatives.

Comparison to McMC-SA. Previous work in layout synthesis does not incorporate constraint gradient information, and instead uses manually crafted McMC moves that augment a layout configuration. While this type of method provides an easier mechanism to explore different layout variations, we achieve the same effect by different initializations.

As observed in our comparison, the run-times of the baseline McMC-SA method increases significantly with the number of layout objects involved. In these approaches, a user can define a more relaxed set of constraints, which at least in principle could lead to a desired layout, but at a much greater computational cost. In practice, we observed that these methods preforms poorly in tightly-packed layouts, leading to unrealistic collisions between layout items. By contrast, our method is dramatically faster due to its continuous nature. Notably, our method requires only a few seconds of run-time for dozens of objects and it naturally scales to hundreds of objects with only moderately increasing computational cost.

Global Optimization Solvers. Since layout synthesis is a nonlinear and nonconvex problem, it is difficult – and fortunately unnecessary – to find the global optimum. We experimented with different non-linear global optimization solvers (NLopt (Johnson, 2011)). Unfortunately, the results were either poor in quality (i.e., unrealistic layouts with many colliding layout items), or the run-times of these methods were intractable. In contrast to these solvers, our method does not directly try to minimize the global energy, but rather satisfies local constraints. Satisfying constraints locally and sequently, results in positional corrections to layout items, that propagate throughout the layout, transforming the layout from an initially deformed state to a converged, satisfactory layout. As in the work of Yu et al. (2011) and Merrell et al. (2011), our approach generates layouts from local minima that satisfy the design constraints, resulting in satisfactory layouts. We use the global energy to evaluate the layout, and for comparing to previous work.

Layout Variety. The intended workflow of layout synthesis methods is that the automated approach generates a variety of viable layouts, from which the user can select the one they most prefer. Our method supports this same workflow. Our method can generate a variety of

layouts by repeatedly running our continuous approach starting from different random initial conditions (Figure 6.11). Different initializations lead to different layout configurations. Additionally, the quality of such a configuration is subjective matter. Hence, at least from the user's perspective, a collision-free global optimum is not definite.

Limitations. Our results are sensitive to initialization, and we need multiple different initialization for an acceptable result. A different initial guess can lead to a different final solution, especially if there are numerous ill defined and conflicting constraints. Experimentally, we found that 5 different initializations is sufficient for most test cases. Additionally, this can also be remedied by adjusting the stiffness of the conflicting constraints, or by defining new constraints that capture a better solution.

We observed that solving constraints sequentially, where the new positions are immediately visible to other constraints, makes quick progress at first, but then the convergence rate slowly decreases as the iterations progress. This method may not converge in the traditional optimization sense; however, our termination criterion is based on the satisfaction of most of the constraints, which is ultimately what matters. For example, in the living-room experiment, the first few iterations yield a layout that is visually similar to the final one, whereas later iterations produce smaller refinements of the layout and resolve cuboid accessibility area intersections. Nevertheless, in practice, we never observed outcomes that failed to satisfy the layout objectives. Like McMC-SA, our energy can increase from one iteration to the next, and gives our method the ability to escape suboptimal local minima. In practice, most starting seeds lead to a satisfactory layout, and all of them lead to a collision free layout. Hence, even though our method is conceptually simple, it yields powerful results.

CHAPTER 7

Conclusion

7.1 Summary

This thesis has proposed a position-based dynamics (PBD) framework for crowd simulation and layout synthesis. Both the agents in crowds and the objects in layouts can be represented as particles and involved in a scenario characterized by general optimization objectives—safely locomoting towards a goal in the case of crowd simulation, or finding a realistic and satisfactory arraignment in the case of layout synthesis. In our framework, the machinery for these goals is provided by different position-based constraints, and the numerical combination of these constraints.

First, we proposed a position-based crowd simulation method that augments position-based dynamics with new positional constraints for simulating agents. These include short and long-range collision avoidance constraints between agents, as well as a modification of existing positional constraints used in the physics-based simulation of effects like friction and cohesion for granular material and fluids. A friction-type constraint between agents allows the formation of pockets and slows down colliding agents, while cohesion allows more uniformity between nearby agents. Each constraint provides positional corrections for the agents. The constraints are combined numerically to compute an agent's position in the next time step. Compared to existing methods, the advantages of a position-based crowd simulation framework are large time steps, guaranteed stability, speed, and ease of control.

Second, we introduced a position-based optimization framework for layout synthesis via positional layout constraints. These layout constraints are geometric and follow well-known interior design guidelines. The optimization starts from a random initialization of layout ob-

jects, and ends when the layout objects are in a non-colliding state satisfying the geometric interior design constraints. We demonstrated our method by synthesizing several layouts, including in sparse and tightly packed settings, from layouts containing several objects to hundreds of objects. Compared to previous work, we were able to synthesize layouts with similar visual quality, but faster by about an order of magnitude. This enabled the fast synthesis of large layouts, which were previously intractable. The main advantage of our method over previous layout synthesis work is speed and scalability, which enables the synthesis of larger and denser layouts than was previously possible.

Our position-based constraint framework allows ease of use and computational efficiency over prior related work. These highly desirable qualities are well-suited to industrial and consumer-facing applications, especially in domains ranging from computer games and visual effects, to computer assisted design software, thus enabling the development of new consumer-oriented products and services in these domains.

7.2 Future Work

7.2.1 Crowd Simulation

Our crowd simulation approach has some limitations. We do not pretend to simulate real pedestrians (cf. (Shao and Terzopoulos, 2007; Yu and Terzopoulos, 2007)). Designing metrics to evaluate such realism is a problem in and of itself, and it is outside the scope of our present work, but we will investigate this topic in future work, including further quantitative analysis of time-to-collision and other anticipatory position analysis. Even though PBD is a simple and stable framework, it requires a certain amount of parameter tuning. We also plan to explore other constraints, such as clamping the magnitude of turning and backward motion of agents. We believe that such a constraint will lead to more realistic results. Finally, experimenting with other online locomotion synthesis methods such as motion fields (Lee et al., 2010) can lead to more interesting agent interactions.

Our position-based framework currently considers only the agent's position and future

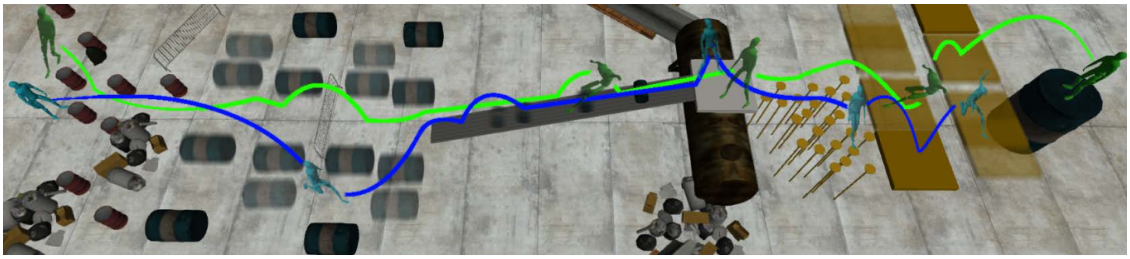


Figure 7.1: Two agents navigating through a complex dynamic environment (Kapadia et al., 2013).

velocity when adjusting the agent's position in the next timestep. Furthermore, the agents are similar in appearance and behavior in everything other than speed and locomotion goals. However, when considering heterogeneous crowds, there are many other appearance and locomotion factors, such as height, size, gait, path, locomotion styles, and other agent-specific parameters, that influence the agent's behavior (Pelechano et al., 2016). Heterogeneous agents of different sizes and gait have different locomotion capabilities, which need to be taken into account (Vaughan and OMalley, 2005). In the future, we would like to design such frameworks for generating more realistic simulation models for heterogeneous crowds. These will be useful in realistic scenarios, as well as for developing future robots that will locomote and work alongside people.

Regarding velocity and acceleration clamping, our current framework depends only on the magnitude of these vectors. This might suffice for simple disc-like Roomba agents, but for human-like agents, velocity and acceleration changes are based not only on the current position (Kwon and Shin, 2007; Zhang et al., 2009). For example, researchers have shown that there are limitations to the curvature of a virtual biped agent's path (Lockwood and Singh, 2011).

Currently, our method employs a simple navigational scheme for planning the agents velocity in the next time step, in which the agents locomote directly to their target. This scheme fails to pass obstacles efficiently, and also fails to consider situations where an agent might prefer taking a longer path that might otherwise be shorter in time (Figure 7.1). Replacing this navigational component with a more realistic dynamical path planning scheme would lead to more realistic simulation results (Kallmann and Kapadia, 2016).

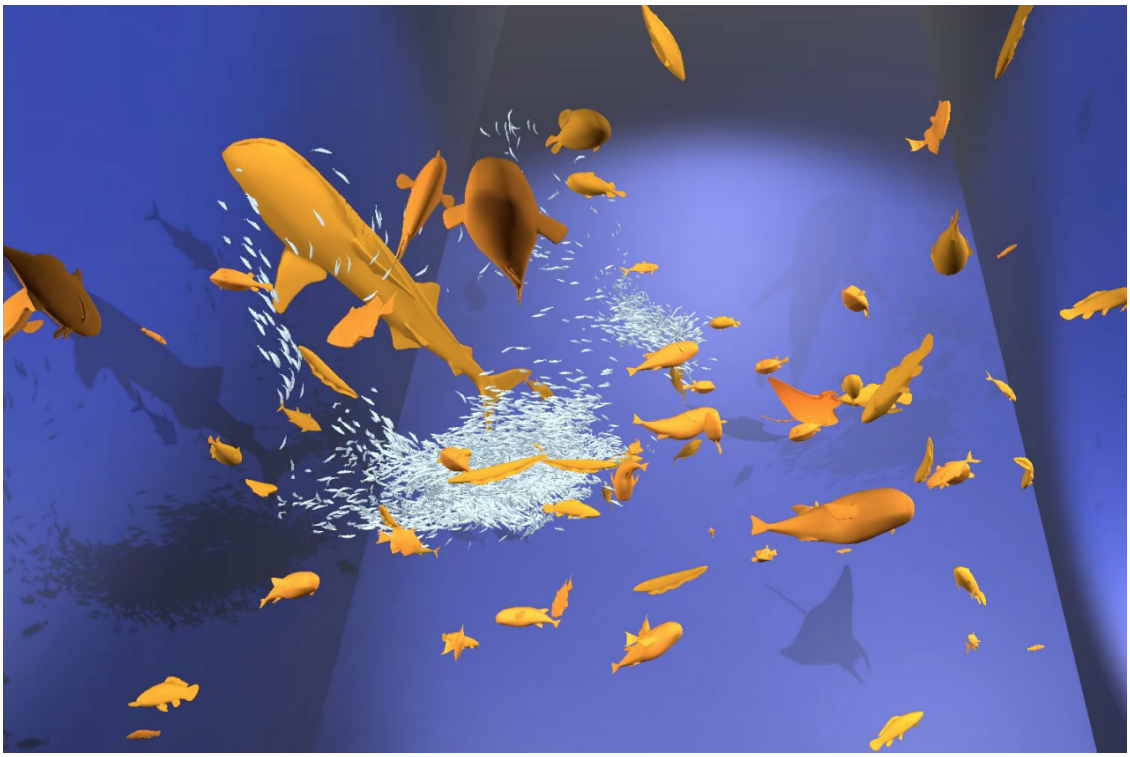


Figure 7.2: Simulating schools of fish avoiding predators (Satoi et al., 2016).

We compared our method to the explicit force-based technique proposed by Karamouzas et al. (2014). In the future, we would like to compare to other popular frameworks such as the social force (Helbing et al., 2007), RVO (Van Den Berg et al., 2011), as well as other recent and concurrent work such as simulating crowds as fluids (Narain et al., 2009), optimization-based approach for crowd simulation and other frameworks (Karamouzas et al., 2017; Wolinski et al., 2016; Dutra et al., 2017).

Our framework provides a collision avoidance mechanism on the 2D plane for bipedal agents. However, extending our framework to 3D will enable the simulation of animals not bound to planar domains, such as fish and birds (Figure 7.2). This is similar to the work demonstrated by Reynolds (1987), however more useful to those who wish to incorporate our method into their already existing PBD simulation software that is commonly used in the visual effects production and game development industries.

In recent years, crowd simulation researchers have designed metrics to evaluate simulation quality, such as the number of colliding agents, evacuation times (or measuring the

effectiveness of the locomotion by the time it takes for agents to get to their locomotion goals), density, entropy, etc. Some crowd simulation methods rely exclusively on these metrics for an agent's navigational and collision avoidance behavior (Narang et al., 2015; Guy et al., 2010b) For completeness, it would be valuable to measure these metrics on scenes simulated with our method.

Finally, since there is little data on the movement of crowds containing more than 10,000 agents (except perhaps evacuations and limited social gatherings), it would also be desirable to design new metrics to asses the quality of a crowd simulation across different settings. Designing such metrics would be also beneficial to the future design of crowd simulation algorithms.

7.2.2 Layout Synthesis

In the present study, we did not encode all the constraints that might be relevant in layout design. However, our method can be easily extended to a broader set of layout constraints in future work. Incorporating GPU parallelization can further speed up the procedure, as could a hierarchical approach, where the layout synthesis problem is broken into stages. It will also be interesting to adjust the stiffness factors in ways that are not merely decreasing or increasing uniformly, in an effort to converge to better global solutions.

As discussed in Section 5.1, an interior layout synthesis is typically nonconvex, which means there is no global minimum solution to be computed straightforwardly. However, there may be a mixed discrete/continuous formulation that might yield more powerful results, similar to a mixed integer-linear programming approach that has recently been proposed for floor plan synthesis (Wu et al., 2018). This is because most furniture arrangements are similar, and they can be defined using graphs and other patterns (Fisher et al., 2011; Xu et al., 2014).

Due to the psuedo-elastic local projection nature of PBD, sometimes there are oscillations and collisions between objects. For example, when there is a collision between the accessibility areas of two objects, the constraint may be partially resolved by projecting one of these

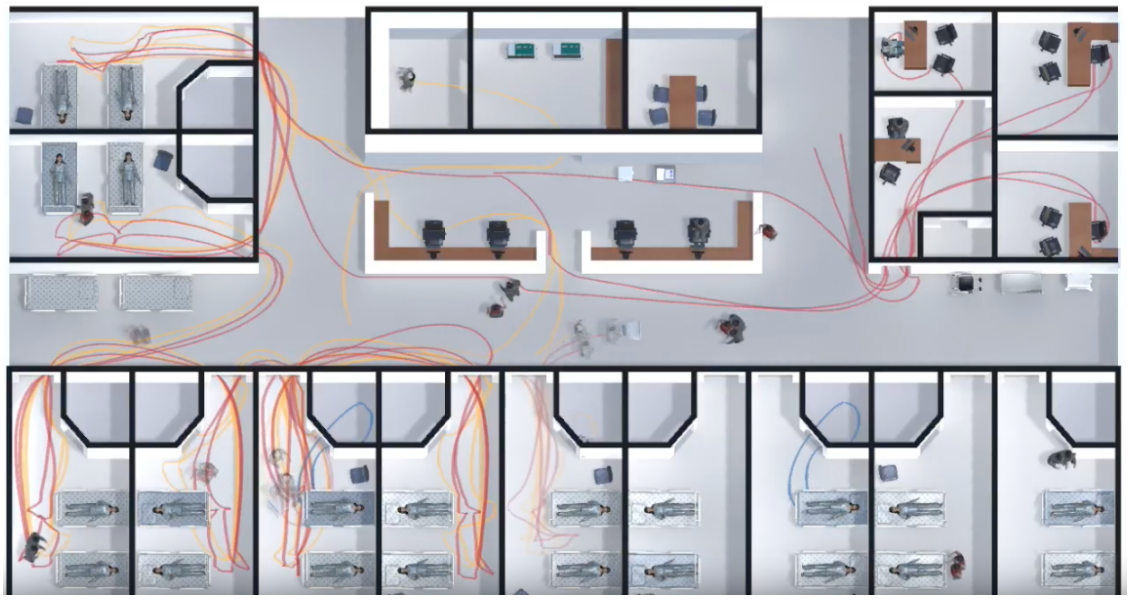


Figure 7.3: Simulating use cases in hospitals using multi-agent narratives (Schaumann et al., 2017). The simulation provides insight into how to better position facilities and rooms for easy access.

objects into a collision with a third object. In future work, we plan to design automatic schemes for detecting and resolving these conditions.

Our current synthesis framework does not allow interactive layout editing and refinement on a large scale. In future work, we would be interested in incorporating a GUI with a touch interface that could support such capabilities, similar to the framework proposed by Yu et al. (2016). Rather than directly editing the layout, a query-based system might be even more helpful to novice users.

Currently, layout constraints must be added manually before the synthesis stage. It would be interesting to automatically infer layout constraints, either from data, or from previous user interactions. Recent work in machine learning and crowd-sourced learning algorithms might provide new insight into these layout constraints, together with an augmented reality layout visualization as described in Appendix C. Most importantly, these constraints might be dynamic and subject to change (Figure 7.3). Hence, combining layout simulation with multiple agents promises to be helpful in assessing the effectiveness of the designed layout (Feng et al., 2016). Finally, our ultimate goal is to design an end-to-end system to synthesize layouts with minimal user intervention.

APPENDIX A

Details of the Constraints Setup

Here in detail are the constraints we used in our experiments.

A.1 Theater Constraints

- Wall distance — stage
- Distance — between each seat and the stage. Each seat is grouped into seating tiers, and each seat receives the corresponding distance constraint of the tier to which it is allocated
- Traffic Lanes — between all the seats and two vectors projected at a symmetric angle form the front face of the stage.
- Orientation — between each seat and the stage. Each seat should face the stage
- Focal point — for each seat in a seating tier. For all seats in a seating tier, the center of mass of that group of chairs should be on the vector projected from the stage's front face.

A.2 Picnic Constraints

- Focal point — Each table is a focal point for a group of 4 chairs
- Distance — between each chair in an associated group
- Distance — the BBQ grills are linked together
- Distance — between each pair of trash cans
- Heat point — between each group of trash cans and a location in the picnic layout

- Heat point — on each table to a different layout area
- Heat point — on the Carousel to the top-middle corner of the layout
- Orientation — between chairs and respective table

A.3 Living-Room Constraints

- Focal point — couch as focal point to table
- Focal point — TV as focal point to couch, sofa chairs
- Focal point — Table as focal point to office chair
- Wall distance and orientation — TV, book case, coat rack, door, indoor plants
- Visual balance
- Orientation — between objects and their respective focal points

A.4 Desk Constraints

- Stacking — between books, divided into two groups
- Heat point — on laptop, to be located near the front middle of the desk.
- Heat point — on notepad to front right of desk
- Heat point — on Rubik's cube to front left of desk
- Distance — between potted plant and book stack
- Distance — between book stack and desk binder
- Distance — between binder and photo frame
- Distance — between photo frame and mug
- Distance — between pencils
- Focal point — laptop as focal point to fruit plate
- Focal point — Rubik's cube as focal point to pencil group
- Focal point — Rubik's cube as a focal point to stack of books
- Wall distance — on one stack of books

A.5 Tightly-Packed Bedroom

- Focal point — Each table is a focal point for a group of 4 chairs
- Distance — between floor lamp, table and chair
- Distance — between chair and table
- Distance — between bookcase and coat rack
- Orientation — between chair and table
- Wall distance — for beds, bookcase and table

A.6 Tightly-Packed Picnic

- Distance — the BBQ grills are linked together
- Distance — between each pair of trash cans
- Heat point — between each group of trash cans and a location in the picnic layout
- Heat point — on the Carousel to the top-middle corner of the layout

APPENDIX B

Simulated Annealing Comparison

This appendix presents all the energy plots from our comparison in the tightly-packed picnic scenario (Figure B.1 and Figure B.2). In each benchmark, the starting positions and orientations are similar, and both simulated annealing and our method use the same optimization parameters.

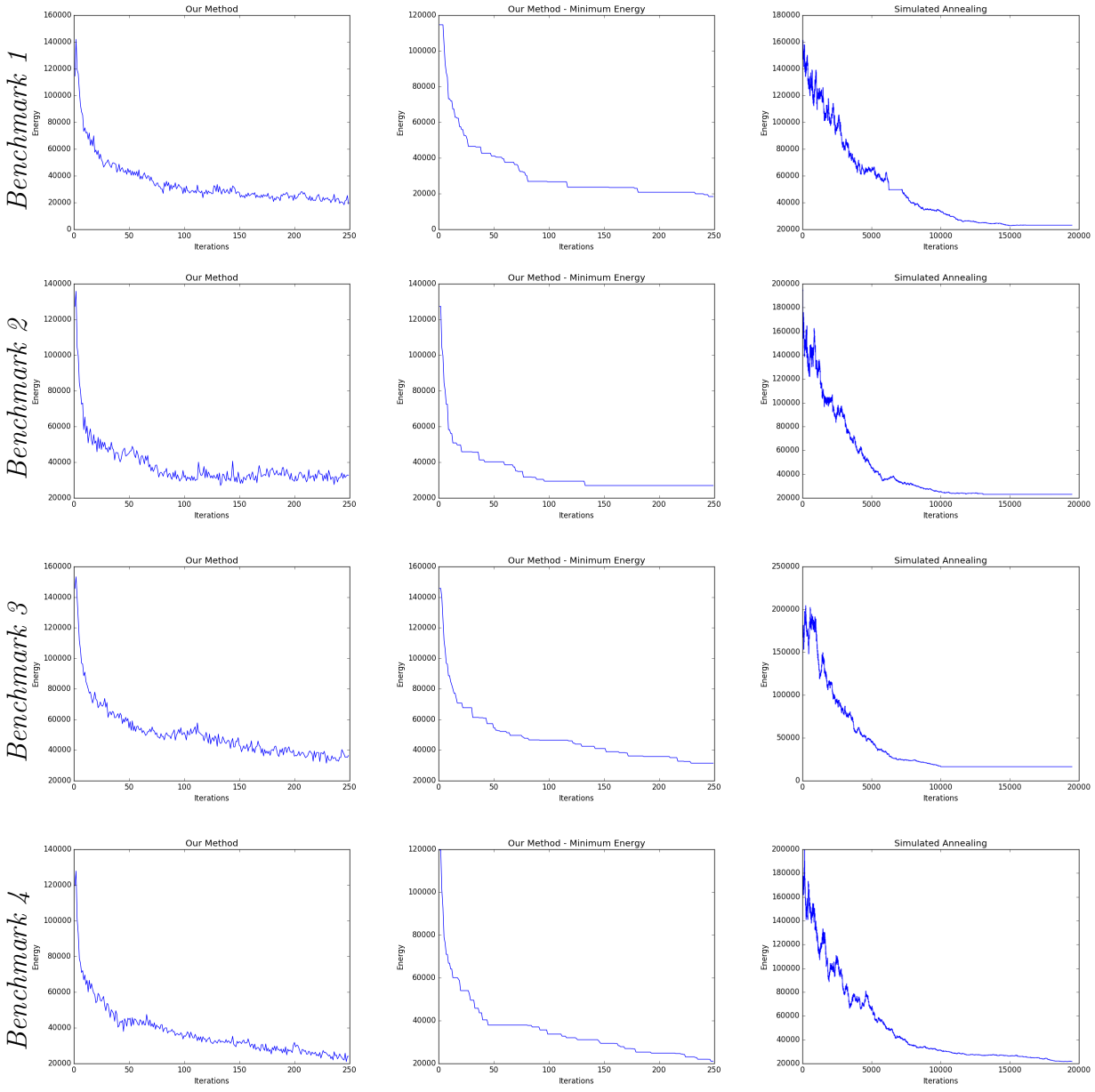


Figure B.1: Energy plots for tightly-packed picnic.

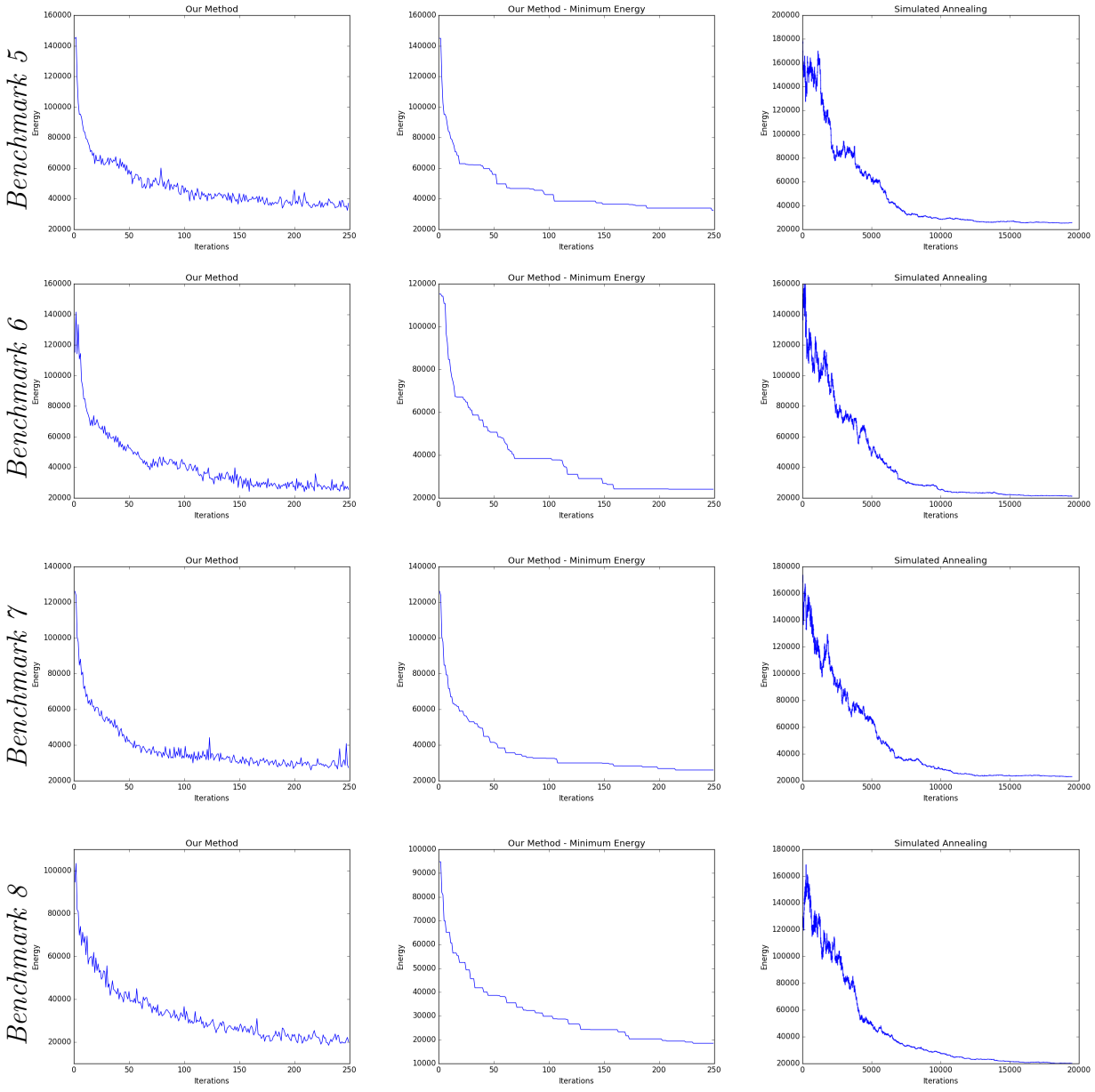


Figure B.2: Energy plots for tightly-packed picnic.

APPENDIX C

Automated Layout Synthesis From Real-World Images

This appendix reproduces the content of our publication (Weiss et al., 2017b).

C.1 Introduction

The arrangement of objects into a layout is an everyday problem. The problem is involved, because a desirable arrangement may vary greatly according to different use cases, individual styles, and other considerations, while various constraints, such as space bounds, the relationships between different objects, as well as comfort and other functional and aesthetic criteria must be enforced. Layout design is far from trivial for people lacking domain experience, as evidenced by the existence of interior design professionals.

Using broadly available smartphones and other camera-enabled mobile devices, it is easy to share photos of indoor or outdoor spaces and receive suggestions from friends or hired professionals on how to organize and furnish the spaces. Popular consumer mobile applications (e.g., by Amazon) provide limited visualizations of selected furnishings using augmented reality. However, prior work has not addressed important aspects of visualizing spaces of interest that incorporate automatically synthesized suggested layouts. A typical use case would be to provide a computerized alternative to the time-consuming process of manually staging a property for sale or lease through the pleasing layout of furniture and other visible accessory items, which can dramatically influence the perceived property value.

Mathematically, layout synthesis yields a challenging non-linear optimization problem. The main goal of our work is to develop a fast, automated system that, given limited user input consisting of a single image of a vacant indoor or outdoor space, visualizes the space

furnished with optimal synthesized layouts. Our approach works well with both interior and certain exterior spaces. It is suitable for implementation as a mobile device application constrained by limited computational resources.

The remainder of this appendix is organized as follows: Section C.2 surveys relevant prior work on layout synthesis as well as on scene understanding from images. Section C.3 overviews our algorithmic approach. Section C.4 presents our results. There follows in Section C.5 a discussion of the limitations of our approach. Section C.6 presents our conclusions and discusses future work.

C.2 Related Work

C.2.1 Layout Synthesis

Layout problems arise in a number of domains. Researchers have applied domain-specific optimization approaches to various layouts, from VLSI layouts (Sarrafzadeh and Lee, 1993) to architectural floor plans (Harada et al., 1995; Merrell et al., 2010). Layout synthesis also appears in the context of generating virtual worlds for computer games, databases for computer vision algorithm testing, and even virtual reality (Smelik et al., 2014; Jiang et al., 2017).

Numerous methods have been proposed for synthesizing layouts. Procedural modeling employs grammars (Müller et al., 2006; Parish and Müller, 2001). However, these methods require a user to manually encode grammatical rules, a complicated task that is similar to implementing a script for a professional modeling package. Graphical user interfaces help simplify this task (Lipp et al., 2008). Additionally, modeling grammars may be augmented to interact with external constraints in the form of guidance shapes, user input, or from other models (Beneš et al., 2011; Šťava et al., 2010; Talton et al., 2011).

Optimization-based methods are used to achieve layout goals under a set of predefined constraints. These methods are usually stochastic in nature, sampling layout arrangements from an unknown probability distribution (Vanegas et al., 2012; Merrell et al., 2011; Yu et al.,

2011; Yeh et al., 2012). However, stochastic methods are usually slow when optimizing a layout with dozens of items. Recent work achieves faster running times by combining or using only a continuous, numerical optimization approach (Weiss et al., 2018; Wu et al., 2018; Bao et al., 2013).

On the consumer side, there exist various software packages and toolkits for designing and visualizing residential layouts (Autodesk, 2011; Reif, 1993). However, these tools require significant manual editing and interior design domain knowledge to achieve satisfactory results.

C.2.2 Understanding Environments

Before augmenting the environment, ideally one must understand the spatial layout and dimensions (Hedau et al., 2009; Gupta et al., 2010), the arrangement of objects within the environment (Jiang et al., 2012a,b; Choi et al., 2015; Jiang et al., 2016), the human influence on these arrangements (Jiang and Saxena, 2013), and where the objects should be placed, from small functional objects (Jiang et al., 2012b) to evaluating the physical quantities of a layout (Zhu et al., 2016).

To understand a scene directly from an image, Ramalingam and Brand (2013) proposed a method for deriving the orientation of indoor and outdoor scenes from a single image, combining vanishing points and an optimization procedure that considers all plausible connectivity constraints between lines. In concurrent work, Izadinia et al. (2017) propose a system that, given an image, reconstructs an approximate virtual replica of the original scene using a database of CAD models and a deep learning framework. Zhang et al. (2016) presented a system for visualizing an augmented indoor scene using a specialized Project Tango tablet, although it does not automatically generate suggestions and requires manual editing.

Algorithm 4 Automatic Layout Method

1: $I \leftarrow$ Get input layout image
2: $O \leftarrow$ Get user layout items and objectives ▷ C.3.3
3: $S \leftarrow$ Segment scene (I) ▷ C.3.1
4: $S \leftarrow$ Estimate 3D Scene (I) ▷ C.3.2
5: $S^* \leftarrow$ Generate Layout Suggestions (S, O)
6: $I^* \leftarrow$ Visualize Augmented Scene (S^*, I) ▷ C.3.3

C.2.3 Pixel-Wise Semantic Segmentation

Deep learning research has grown dramatically in recent years thanks to algorithmic advances combined with efficient and powerful implementations on GPUs. Most recent results are based on the Visual Geometry Group (VGG) network proposed by Simonyan and Zisserman (2014), a very deep network that has produced state of art accuracy in image classification tasks, with various modifications. FCN (Long et al., 2015), DeepLab (Chen et al., 2016), and Dilated Convolutions (Yu and Koltun, 2016) perform pixel-wise semantic segmentation and have yielded good accuracy for such segmentation problems. The benchmark for semantic segmentation algorithms is the Pascal dataset, which contains images from various domains. Recently, another pixel-wise semantic segmentation algorithm was proposed by Badrinarayanan et al. (2015), which also employed a VGG net architecture. In addition to the newly proposed techniques, the network was trained on both a road dataset and an indoor scene dataset provided by SUN-RGBD (Xiao et al., 2010).

C.3 Algorithm

The initial input to our method is an ordinary image of an indoor or outdoor environment. First, we semantically segment the scene depicted in the image, separating the floor/ground from other objects in the scene. Second, we detect the boundary of the floor and other significant edge features. Third, we measure the scale and orientation of the scene by using a checkerboard calibration marker. Finally, we generate an optimal layout with user-selected layout items. Algorithm 4 provides an overview of our approach, indicating the sections in which the details of each step are presented.

C.3.1 Semantic Segmentation of the Scene

We use a pixel-wise semantic segmentation algorithm to extract the floor/ground. There exist various algorithmic approaches for this pixel-wise segmentation. We chose to use SegNet (Badrinarayanan et al., 2015), whose model is trained with a SUN-RGBD dataset (Xiao et al., 2010), mainly because this dataset contains only indoor scene objects while most other datasets have a mix of images from various other categories. Badrinarayanan et al. (2015) trained their network with 37 categories, including common layout objects, such as walls, chairs, tables, and the floors of indoor scenes.

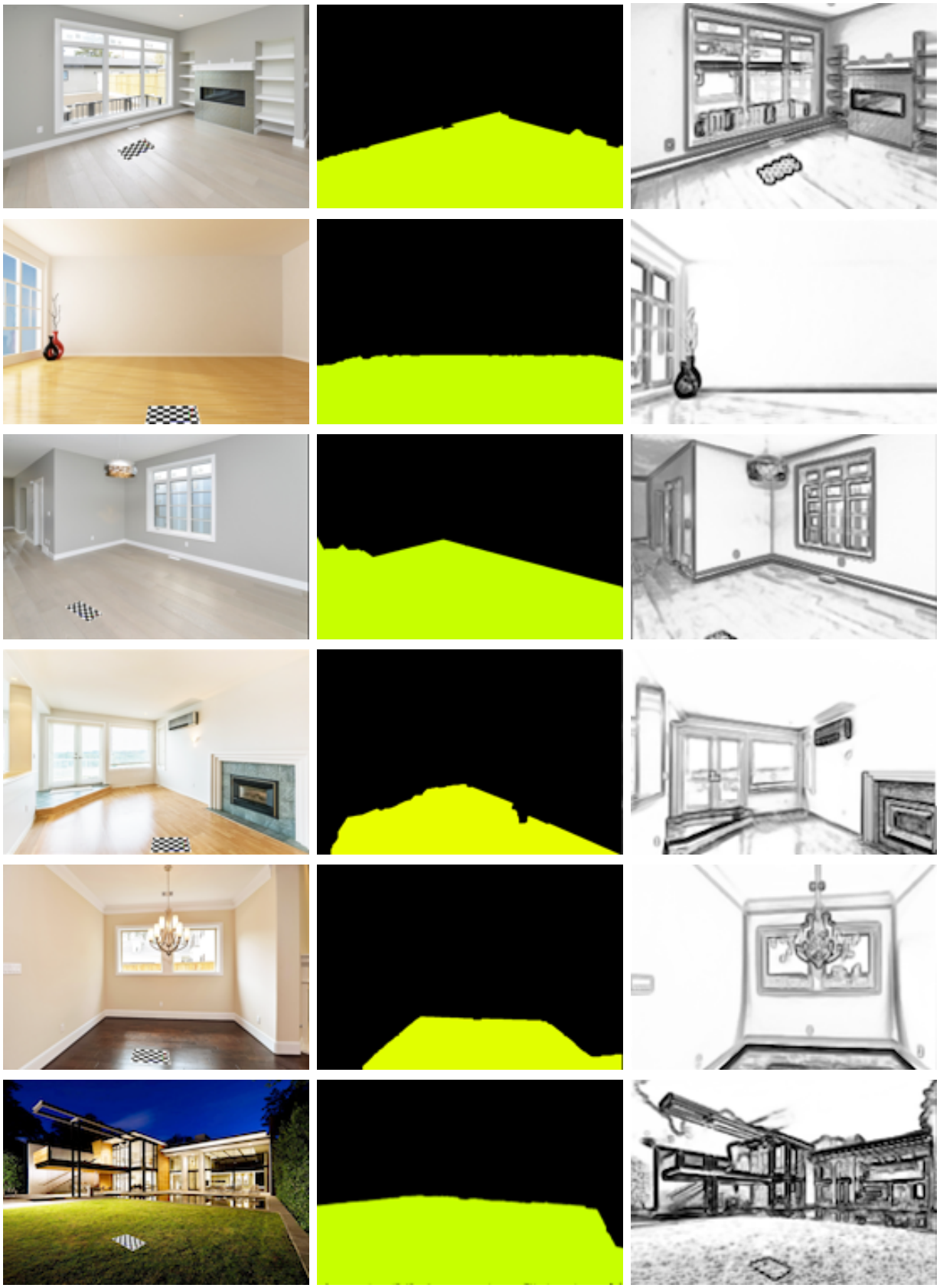
For synthesizing layouts, we first must estimate the ground area available. To that end, we need to segment the floor and remove irrelevant objects from the scene. Thus, our task is a binary segmentation problem rather than a multi-label segmentation one. To achieve this segmentation, we add one more layer after the softmax layer of VGG net. This layer takes its output from the softmax layer and produces binary class labels for the floor and other parts of the image. This output is represented by a simple conditional statement where the 'floor' class from the softmax layer produces 'floor', and all of the other labels become 'others'.

Subsequent to the pixel-wise segmentation, we use the approach suggested in GrabCut (Rother et al., 2004) to retain the floor and remove other components of the scene. Figure C.1 shows the result of this segmentation process. Thus, we detect the location and boundary of the floor or ground, which is necessary in order to establish a layout plane upon which we will synthesize layout items.

C.3.2 Inferring a 3D Estimate of the Scene

To estimate the size of the room, we ask users to place a checkerboard calibration marker in the scene. The scale of the room is estimated by comparing the known size of the checkerboard to the segmented floor in the scene. The details of this process are as follows:

1. Detect the checkerboard in the scene using Harris corner detection, and compute the room's width, height, and center in pixel coordinates.



(a) (b) (c)

Figure C.1: The results of our segmentations and edge detection. (a) Original images. (b) Segmented floors/ground. (c) Edge maps.

(a) (b) (c)

Figure C.2: Synthesized game-room layout. (a) random initial state. (b) Intermediate state of the optimization; layout items are too close for the layout to be deemed comfortable. (c) Final suggested layout with relaxed spacing.

2. Define an origin at the center of the checkerboard on the x and y axes of the plane to the horizontal and vertical directions, respectively. The z axis is determined as the cross product of the x and y axes.

3. Employ the checkerboard to calibrate the camera and set up the internal parameters. The camera pose is estimated based on the distance and orientation from the origin. This step is necessary to estimate the position of the virtual camera when rendering the final layout containing the virtual 3D furniture.

4. Traverse the floor geometry from the origin along the x and y axes to compute the distance from the origin to the edge of the floor in pixel coordinates. Since we know the ratio between the size of the checkerboard in pixel coordinates and its real size, we can compute the length of the floor in a scene by scaling its length in pixel coordinates by the ratio obtained from the checkerboard.

5. Apply the Holistically-Nested Edge Detection algorithm (Xie and Tu, 2015) to detect the edges of the scene (Figure C.1(c)). This deep-learning-based edge detector results in better edge detection for our indoor scene images than traditional edge detectors, such as the Canny edge detector. We search for edges that are aligned to the z axis, by selecting edge vectors $\mathbf{l}$ whose cosine distance to $\mathbf{z}$,

$$\cos(\mathbf{z}, \mathbf{l}) = \frac{\mathbf{z} \cdot \mathbf{l}}{||\mathbf{z}||\ ||\mathbf{l}||}, \tag{C.1}$$

is greater than threshold $t = 0.9$. We use the longest edge as the height of the space.

C.3.3 Layout Synthesis and Visualization

Our layout synthesis scheme is based on a continuous numerical approach to layout synthesis, which was inspired by Position-Based Dynamics (Müller et al., 2007) and by a stochastic McMC scheme for optimizing indoor layouts (Yu et al., 2011).

Given a layout, the user specifies the furniture items to arrange, and the objectives of that arrangement. The objectives are annotated in terms of geometric constraints, similar to those described in recent work (Yu et al., 2011; Merrell et al., 2011). Among other constraints, our method supports pairwise distance, distance to wall, pairwise and wall rotational constraints, and visual balance, and it can easily be extended with additional constraints. In a typical use case, a user can define the distance between layout items in the same group, together with distance and orientation to the nearest wall. Collisions between competing layout items are automatically resolved. We refer the reader to the above cited papers for the details.

While Weiss et al. (2018), Yu et al. (2011), and Merrell et al. (2011) generate suggestions on a per-object level of detail, we speed up layout generation in some cases by employing a rule-based approach in which each object encompasses a group of layout items. For example, a dining table (Figure C.2) is usually accompanied by a set of matching chairs at prescribed distances and orientations. This is most apparent when furnishing an empty layout. In the common use case, a user who is interested in furnishing an interior layout travels to a nearby retail store and buys furniture items in combinations, as suggested by the retail catalog or displays on site; e.g., a collection that includes sofas, adjoining sofa chairs, and a coffee table.

After obtaining the input layout area, input layout items, and constraints between these items, our method synthesizes the layout, which results in a 3D scene. Given the estimated camera pose, the 3D representation of the layout is rendered into the original image I, to yield the augmented image I^*. For rendering, we use Blender Cycles with the same settings for every scene, except for varying the exposure setting to produce a more realistic result by matching the illumination in I.

(a) (b) (c)

Figure C.3: (a) A vacant bedroom. (b) Intermediate layout. (c) Synthesized bedroom.

(a) (b) (c)

Figure C.4: Living Room. Our method provides different layout suggestions (a)–(c) for the same collection of furniture items and layout constraints.

C.4 Results

Our framework is composed of two main components. The semantic segmentation and edge detection component is implemented in Python on an Ubuntu machine with a 3.4 GHz Intel Core i7 and Nvidia Titan X GPU. We use Caffe (Jia et al., 2014), one of the most common libraries for deep learning in computer vision, for both image processing tasks. Semantic segmentation and edge detection take on average approximately 0.1 seconds and 0.5 seconds, respectively. The layout synthesis component is implemented in Python and Cython. Each layout suggestion is synthesized in no more than 4 seconds. Rendering the final result takes approximately 3 seconds.

We briefly summarize our experiments next:

Hosting Room: (Figure 6.14) We used a set of 5 dining tables, a clothes rack, and a floor lamp. We did not impose any strict distance constraints in this setting, except for the floor lamp to be close to the wall.

Outdoor Yard: (Figure 6.15) In this scenario, we assigned two patio-style tables with chairs, where the chairs are constrained to be around the tables. We also added a BBQ grill and a trash can. The grill is constrained to be at a greater than minimal distance from the other layout items.

Game Room: (Figure C.2) This scenario includes a sofa, floor lamp, two table tennis tables, and two dining tables for socializing. The sofa is constrained to be near the wall, and the tennis tables close to each other. The sofa is positioned in a slightly rotated position relative to the room's wall for a more comfortable interaction.

Bedroom: (Figure C.3) The bedroom setting included a bed, closet, clothes rack, office table, chair, and a floor lamp. The bed and closet were constrained to be near the wall, and the floor lamp near the table.

Living Room: (Figure C.4) We experimented with a typical living room layout, consisting of a TV, sofa, sofa chairs, and coffee table. We also added an extra table and office chair, and plants. In this setting, we constrained the TV to be the focal point of the furniture groups, consisting of the sofa, sofa chairs, and coffee table. Our system generated 3 different layout suggestions for the same layout items and constraints.

C.5 Discussion

We have demonstrated the efficacy of our method by augmenting various input scenes, both indoor and outdoor. However, our method has some limitations. An incorrect semantic segmentation of the scene is possible. This typically happens when the input image is not clean; e.g., it contains a large cast shadow, is under-exposed or over-exposed, or the floor has a non-uniform pattern. Fortunately, these are unlikely occurrences for our target use case; i.e., real estate staging, since the sellers of a property tend to capture high-quality images.

In the present study, we did not train our network on outdoor scenes. Therefore, we expect these scenes to be more challenging. Nevertheless, we also tested our method on outdoor scenes (Figure 6.15), obtaining some adequate results. We observed that outdoor

scenes where similar to indoor scenes with respect to the surface layouts and other segmentation features of the latter. In general, however, our method will not be reliable on outdoor images.

We used a checkerboard to determine the scale of the scene, calibrate the camera, and appropriately set the virtual camera used to render the completed layout. This calibration worked well when the space is approximately rectangular, but it is not accurate in cases where the space has a non-standard shape. Moreover, placing a marker in a scene is not always convenient.

A user of our method must manually assign constraints for the layout synthesis step. Assigning these constraints is straightforward, albeit not automatic. This step can easily be interchanged with a user-friendly set of questions regarding the user's layout preferences.

C.6 Conclusion and Future Work

To our knowledge, our system is the first complete, interactive system for augmenting images of indoor or outdoor spaces with the highly automated synthesis of furnished layouts. Users of our system can range from ordinary consumers who are looking for a new residence or are interested in remodeling an existing residence, to interior designers and real-estate professionals.

In future work, we plan to improve our system by training it on a more diverse set of images; e.g., by collecting various images, including outdoor scenes and more complex or cluttered interior scenes from the several commonly available datasets (Everingham et al., 2015; Song et al., 2015; Silberman et al., 2012; Jiang et al., 2017). Additionally, to further improve the accuracy of our system, we plan to implement a better scene layout understanding algorithm, such as the one by Ramalingam and Brand (2013) that estimates a layout using vanishing lines, or the one by Ren et al. (2016a) that proposes FCN-based scene layout estimation. We also plan to combine holistically-nested edge detection (Xie and Tu, 2015) to improve the accuracy of our layout detection, which should provide us better 3D scene understanding. Better scene understanding would enable an extended version of our system to

handle images of spaces containing existing furniture, which would help people who want to add new pieces of furniture or otherwise augment their spaces. Finally, we are interested in improving visual quality by incorporating an effective approach for estimating the lightning conditions in the original image (Karsch et al., 2011), so as to more realistically illuminate the synthesized layouts.

Algower, E. L. and Georg, K. (2003). *Introduction to Numerical Continuation Methods*, volume 45 of *SIAM Classics in Applied Mathematics*. SIAM.

Arvin, S. A. and House, D. H. (2002). Modeling architectural design objectives in physically based space planning. *Automation in Construction*, 11(2):213–225.

Attar, R., Aish, R., Stam, J., Brinsmead, D., Tessier, A., Glueck, M., and Khan, A. (2009). Physics-based generative design. In *CAAD Futures Conf.*, pages 231–244.

Autodesk (2011). Homestyler. http://www.homestyler.com.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*.

Bao, F., Yan, D.-M., Mitra, N. J., and Wonka, P. (2013). Generating and exploring good building layouts. *ACM Trans. Graph.*, 32(4).

Baraff, D. and Witkin, A. (1998). Large steps in cloth simulation. In *Proc. SIGGRAPH*, SIGGRAPH '98, pages 43–54.

Bender, J., Koschier, D., Charrier, P., and Weber, D. (2014a). Position-based simulation of continuous materials. *Comp. Graph.*, 44:1–10.

Bender, J., Müller, M., Otaduy, M., Teschner, M., and Macklin, M. (2014b). A survey on position-based simulation methods in computer graphics. *Comp. Graph. Forum.*, 33(6):228–251.

Beneš, B., Šťava, O., Měch, R., and Miller, G. (2011). Guided procedural modeling. *Comp. Graph. Forum.*, 30(2):325–334.

Bouaziz, S., Martin, S., Liu, T., Kavan, L., and Pauly, M. (2014a). Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.*, 33(4):154:1–154:11.

Bouaziz, S., Martin, S., Liu, T., Kavan, L., and Pauly, M. (2014b). Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.*, 33(4):154.

Bruneau, J. and Pettré, J. (2015). Energy-efficient mid-term strategies for collision avoidance in crowd simulation. In *Symp. Comp. Anim.*, pages 119–127.

Cao, Y., Chan, A. B., and Lau, R. W. (2012). Automatic stylistic manga layout. *ACM Trans. Graph.*, 31(6):141.

Cao, Y., Lau, R. W., and Chan, A. B. (2014). Look over here: Attention-directing composition of manga elements. *ACM Trans. Graph.*, 33(4):94.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2016). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *arXiv Preprint arXiv:1606.00915*.

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

Choi, W., Chao, Y.-W., Pantofaru, C., and Savarese, S. (2015). Indoor scene understanding with geometric and semantic contexts. *Int. J. Comp. Vis. (IJCV)*, pages 204–220.

Cox, T. and Cox, M. (2000). *Multidimensional Scaling, Second Edition*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. CRC Press.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Comp. Vis. and Pat. Recognition (CVPR)*. IEEE.

Deasy, C. and Lasswell, T. E. (1990). *Designing Places for People*. Whitney.

DeChiara, J., Panero, J., and Zelnik, M. (2001). *Time-Saver Standards for Interior Design and Space Planning*. Time-Saver Standards. McGraw-Hill Education.

Desbrun, M., Schröder, P., and Barr, A. (1999). Interactive animation of structured deformable objects. In *Proc. of Graph. Interface*, pages 1–8.

Dutra, T. B., Marques, R., Cavalcante-Neto, J. B., Vidal, C. A., and Pettré, J. (2017). Gradient-based steering for vision-based crowd simulation algorithms. *Comp. Graph. Forum.*, 36(2):337–348.

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2015). The Pascal visual object classes challenge: A retrospective. *Int. J. Comp. Vis. (IJCV)*, 111(1):98–136.

Feng, T., Yu, L.-F., Yeung, S.-K., Yin, K., and Zhou, K. (2016). Crowd-driven mid-scale layout design. *ACM Trans. Graph.*, 35(4).

Fisher, M., Ritchie, D., Savva, M., Funkhouser, T., and Hanrahan, P. (2012). Example-based synthesis of 3D object arrangements. *ACM Trans. Graph.*, 31(6):135.

Fisher, M., Savva, M., and Hanrahan, P. (2011). Characterizing structural relationships in scenes using graph kernels. *ACM Trans. Graph.*, 30(4):34.

Fratarcangeli, M. and Pellacini, F. (2015). Scalable partitioning for parallel position based dynamics. *Comp. Graph. Forum.*, 34(2):405–413.

Golas, A., Narain, R., and Lin, M. (2013). hybrid long-range collision avoidance for crowd simulation. In *Symp. I3D Graph. Games*, I3D '13, pages 29–36.

Golas, A., Narain, R., and Lin, M. (2014). Continuum modeling of crowd turbulence. *Phys Rev E*, 90:042816.

Golub, G. H. and Varga, R. S. (1961). Chebyshev semi-iterative methods, successive over-relaxation iterative methods, and second order richardson iterative methods. *Numerische Mathematik*, 3(1):157–168.

Green, S. (2008). Cuda particles. *NVIDIA Whitepaper*, 2(3.2).

Grosso, P. B. (1985). *Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model*. PhD thesis, University of Michigan, Ann Arbor, MI, USA.

Gupta, A., Hebert, M., Kanade, T., and Blei, D. M. (2010). Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *Adv. Neural Info. Proc. Sys. (NIPS)*, pages 1288–1296.

Guy, S., Chhugani, J., Curtis, S., Dubey, P., Lin, M., and Manocha, D. (2010a). Pledestrians: A least-effort approach to crowd simulation. In *Symp. Comp. Anim.*, SCA '10, pages 119–128.

Guy, S., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., and Dubey, P. (2009). Clearpath: Highly parallel collision avoidance for multi-agent simulation. In *Symp. Comp. Anim.*, SCA '09, pages 177–187.

Guy, S. J., Chhugani, J., Curtis, S., Dubey, P., Lin, M., and Manocha, D. (2010b). Pledestrians: a least-effort approach to crowd simulation. In *Symp. Comp. Anim.*, SCA '10, pages 119–128. Eurographics Association.

Harada, M., Witkin, A., and Baraff, D. (1995). Interactive physically-based manipulation of discrete/continuous models. *ACM Trans. Graph.*, pages 199–208.

He, L., Pan, J., Narang, S., and Manocha, D. (2016). Dynamic group behaviors for interactive crowd simulation. In *Symp. Comp. Anim.*, pages 139–147.

Hedau, V., Hoiem, D., and Forsyth, D. (2009). Recovering the spatial layout of cluttered rooms. In *Proc. Int. Conf. on Comp. Vis. (ICCV)*.

Helbing, D., Farkas, I., and Vicsek, T. (2000). Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490.

Helbing, D., Johansson, A., and Al-Abideen, H. (2007). Dynamics of crowd disasters: An empirical study. *Phys Rev E*, 75(4):046109.

Helbing, D. and Molnar, P. (1995). Social force model for pedestrian dynamics. *Phys Rev E*, 51(5):4282.

Hua, B.-S., Pham, Q.-H., Nguyen, D. T., Tran, M.-K., Yu, L.-F., and Yeung, S.-K. (2016). Scenenn: A scene meshes dataset with annotations. In *3D Vision (3DV)*, pages 92–101. IEEE.

Izadinia, H., Shan, Q., and Seitz, S. M. (2017). IM2CAD. In *IEEE Conf. Comp. Vis. Pat. Rec. (CVPR)*.

James, D. L. and Pai, D. K. (1999). Artdefo: accurate real time deformable objects. In *Proc. of Comp. Graph. and Interactive Techniques*, pages 65–72. ACM Press/Addison-Wesley Publishing Co.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proc. ACM Int. Conf. Multimedia*, pages 675–678.

Jiang, C., Zhu, Y., Qi, S., Huang, S., Lin, J., Guo, X., Yu, L.-F., Terzopoulos, D., and Zhu, S.-C. (2017). Configurable, Photorealistic Image Rendering and Ground Truth Synthesis by Sampling Stochastic Grammars Representing Indoor Scenes. *arXiv Preprint arXiv:1704.00112*.

Jiang, H., Xu, W., Mao, T., Li, C., Xia, S., and Wang, Z. (2010). Continuum crowd simulation in complex environments. *Comp. Graph.*, 34(5):537 – 544.

Jiang, Y., Koppula, H. S., and Saxena, A. (2016). Modeling 3D environments through hidden human context. *IEEE Trans. Pat. Analysis and Machine Intelligence*, pages 2040–2053.

Jiang, Y., Lim, M., and Saxena, A. (2012a). Learning object arrangements in 3D scenes using human context. In *Proc. Int. Conf. on Machine Learning*, USA.

Jiang, Y., Lim, M., Zheng, C., and Saxena, A. (2012b). Learning to place new objects in a scene. *Int. J. Robotics Research*, 31(9):1021–1043.

Jiang, Y. and Saxena, A. (2013). Infinite latent conditional random fields for modeling environments through humans. In *Robotics: Science and Systems*, pages 1–8.

Johnson, S. G. (2011). *The NLopt nonlinear-optimization package*.

Jones, L. M. and Allen, P. S. (2014). *Beginnings of Interior Environments*. Pearson.

Kallmann, M. and Kapadia, M. (2016). Geometric and discrete path planning for interactive virtual worlds. *Synthesis Lectures on Visual Computing: Computer Graphics, Animation, Computational Photography, and Imaging*, 8(1):1–201.

Kapadia, M., Beacco, A., Garcia, F., Reddy, V., Pelechano, N., and Badler, N. I. (2013). Multi-domain real-time planning in dynamic environments. In *Symp. Comp. Anim.*, pages 115–124. ACM.

Karamouzas, I., Skinner, B., and Guy, S. (2014). Universal power law governing pedestrian interactions. *Phys Rev Lett*, 113:238701.

Karamouzas, I., Sohre, N., Narain, R., and Guy, S. (2017). Implicit crowds: Optimization integrator for robust crowd simulation. *ACM Trans. Graph.*, 36(4).

Karsch, K., Hedau, V., Forsyth, D., and Hoiem, D. (2011). Rendering synthetic objects into legacy photographs. *ACM Trans. Graph.*, 30(6):157.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *J. Statistical Phys.*, 34(5-6):975–986.

Kwon, T. and Shin, S. Y. (2007). A steering model for on-line locomotion synthesis. *Computer Animation and Virtual Worlds*, 18:463–472.

Lee, Y., Wampler, K., Bernstein, G., Popović, J., and Popović, Z. (2010). Motion fields for interactive character locomotion. *ACM Trans. Graph.*, 29(6):138.

Lipp, M., Wonka, P., and Wimmer, M. (2008). Interactive visual editing of grammars for procedural architecture. *ACM Trans. Graph.*, 27(3).

Liu, T., Bargteil, A., O'Brien, J., and Kavan, L. (2013). Fast simulation of mass-spring systems. *ACM Trans. Graph.*, 32(6):209:1–7.

Lockwood, N. and Singh, K. (2011). Biomechanically-inspired motion path editing. In *Symp. Comp. Anim.*, pages 267–276. ACM.

Lok, S., Feiner, S., and Ngai, G. (2004). Evaluation of visual balance for automated layout. In *Proc. of the 9th Int. Conf. on Intelligent user interfaces*, pages 101–108. ACM.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *IEEE Conf. Comp. Vis. Pat. Rec. (CVPR)*, pages 3431–3440.

Macklin, M. and Müller, M. (2013). Position based fluids. *ACM Trans. Graph.*, 32(4):104:1–104:12.

Macklin, M., Müller, M., and Chentanez, N. (2016). Xpbd: Position-based simulation of compliant constrained dynamics. In *Motion in Games*, MIG '16, pages 49–54.

Macklin, M., Müller, M., Chentanez, N., and Kim, T. (2014a). Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33(4):153:1–153:12.

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

Majerowicz, L., Shamir, A., Sheffer, A., and Hoos, H. H. (2014). Filling your shelves: Synthesizing diverse style-preserving artifact arrangements. *IEEE Trans. Vis. Comp. Graph.*, 20(11):1507–1518.

Manteaux, P.-L., Wojtan, C., Narain, R., Redon, S., Faure, F., and Cani, M.-P. (2016). Adaptive physically based models in computer graphics. In *Comp. Graph. Forum.* Wiley Online Library.

Merrell, P., Schkufza, E., and Koltun, V. (2010). Computer-generated residential building layouts. In *ACM Trans. Graph.*, volume 29, page 181. ACM.

Merrell, P., Schkufza, E., Li, Z., Agrawala, M., and Koltun, V. (2011). Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.*, 30(4):87.

Müller, M., Chentanez, N., Kim, T.-Y., and Macklin, M. (2014). Strain based dynamics. In *Symp. Comp. Anim.*, pages 149–157.

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

Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. (2007). Position based dynamics. *J Vis Comm Imag Repre*, 18(2):109–118.

Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L. (2006). Procedural modeling of buildings. *ACM Trans. Graph.*, 25(3).

Narain, R., Golas, A., Curtis, S., and Lin, M. (2009). Aggregate dynamics for dense crowd simulation. *ACM Trans. Graph.*, 28(5):122:1–122:8.

Narain, R., Overby, M., and Brown, G. E. (2016). Admm $\supseteq$ projective dynamics: Fast simulation of general constitutive models. In *Symp. Comp. Anim.*, pages 21–28. Eurographics Association.

Narang, S., Best, A., Curtis, S., and Manocha, D. (2015). Generating pedestrian trajectories consistent with the fundamental diagram based on physiological and psychological factors. *PLoS one*, 10(4):e0117856.

O'brien, J. F. and Hodgins, J. K. (1999). Graphical modeling and animation of brittle fracture. In *Proc. of Comp. Graph. and Interactive Techniques*, pages 137–146. ACM Press/Addison-Wesley Publishing Co.

Parish, Y. I. H. and Müller, P. (2001). Procedural modeling of cities. In *Proc. SIGGRAPH*, pages 301–308.

Pelechano, N., Allbeck, J. M., Kapadia, M., and Badler, N. I. (2016). *Simulating heterogeneous crowds with interactive behaviors*. CRC Press.

Peng, C.-H., Yang, Y.-L., and Wonka, P. (2014). Computing layouts with deformable templates. *ACM Trans. Graph.*, 33(4):99.

Qin, H. and Terzopoulos, D. (1996). D-nurbs: a physics-based framework for geometric design. *IEEE Trans. Vis. Comp. Graph.*, 2(1):85–96.

Ramalingam, S. and Brand, M. (2013). Lifting 3D Manhattan lines from a single image. In *Proc. Int. Conf. on Comp. Vis. (ICCV)*, pages 497–504.

Reif, D. (1993). *Home Quick Planner: Reusable, Peel & Stick Furniture & Architectural Symbols*. Gardeners' Guide.

Reinert, B., Ritschel, T., and Seidel, H.-P. (2013). Interactive by-example design of artistic packing layouts. *ACM Trans. Graph.*, 32(6):218.

Ren, Y., Chen, C., Li, S., and Kuo, C.-C. J. (2016a). A coarse-to-fine indoor layout estimation (CFILE) method. *arXiv Preprint arXiv:1607.00598*.

Ren, Z., Charalambous, P., Bruneau, J., Peng, Q., and Pettré, J. (2016b). Group modeling: A unified velocity-based approach. In *Comp. Graph. Forum.* Wiley Online Library.

Reynolds, C. (1987). Flocks, herds and schools: A distributed behavioral model. *Comp. Graph.*, 21(4):25–34.

Reynolds, C. (1999). Steering behaviors for autonomous characters. In *Game Dev Conf*, volume 1999, pages 763–782.

Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314.

Sarrafzadeh, M. and Lee, D.-T. (1993). *Algorithmic Aspects of VLSI Layout*, volume 2. World Scientific.

Satoi, D., Hagiwara, M., Uemoto, A., Nakadai, H., and Hoshino, J. (2016). Unified motion planner for fishes with various swimming styles. *ACM Trans. Graph.*, 35(4).

Schaumann, D., Breslav, S., Goldstein, R., Khan, A., and Kalay, Y. E. (2017). Simulating use scenarios in hospitals using multi-agent narratives. *J. of Building Performance Simulation*, 10(5-6):636–652.

Schechter, H. and Bridson, R. (2012). Ghost sph for animating water. *ACM Trans. Graph.*, 31(4):61:1–61:8.

Shao, W. and Terzopoulos, D. (2007). Autonomous pedestrians. *Graph Models*, 69(5-6):246–274.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. *European Conf. on Comp. Vis. (ECCV)*, pages 746–760.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv Preprint arXiv:1409.1556*.

Smelik, R. M., Tutenel, T., Bidarra, R., and Benes, B. (2014). A survey on procedural modelling for virtual worlds. *Comp. Graph. Forum.*, 33(6):31–50.

Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun RGB-D: A RGB-D scene understanding benchmark suite. In *IEEE Conf. Comp. Vis. Pat. Rec. (CVPR)*, pages 567–576.

Stam, J. (2009). Nucleus: Towards a unified dynamics solver for computer graphics. In *Comp Design Comp Graph.*, pages 1–11. IEEE.

Šťava, O., Beneš, B., Měch, R., Aliaga, D. G., and Krištof, P. (2010). Inverse procedural modeling by automatic generation of L-systems. *Comp. Graph. Forum.*, 29(2).

Talbott, C., Matthews, M., and Cosentino, C. (1999). *Decorating for Good: A Step-by-step Guide to Rearranging what You Already Own*. C. Potter.

Talton, J. O., Lou, Y., Lesser, S., Duke, J., Měch, R., and Koltun, V. (2011). Metropolis procedural modeling. *ACM Trans. Graph.*, 30(2).

Teran, J., Blemker, S., Hing, V., and Fedkiw, R. (2003). Finite volume methods for the simulation of skeletal muscle. In *Symp. Comp. Anim.*, SCA '03, pages 68–74. Eurographics Association.

Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. (1987). Elastically deformable models. *ACM Trans. Graph.*, 21(4):205–214.

Teschner, M., Heidelberger, B., Muller, M., and Gross, M. (2004). A versatile and robust model for geometrically complex deformable solids. In *Proc. of Comp. Graph. International*, pages 312–319. IEEE.

Teschner, M., Heidelberger, B., Müller, M., Pomerantes, D., and Gross, M. H. (2003). Optimized spatial hashing for collision detection of deformable objects. In *VMV*, volume 3, pages 47–54.

Thalmann, D., Grillon, H., Maim, J., and Yersin, B. (2009). Challenges in crowd simulation. In *Int Conf on CyberWorlds*, pages 1–12. IEEE.

Tonge, R., Benevolenski, F., and Voroshilov, A. (2012). Mass splitting for jitter-free parallel rigid body simulation. *ACM Trans. Graph.*, 31(4):105.

Treuille, A., Cooper, S., and Popović, Z. (2006). Continuum crowds. *ACM Trans. Graph.*, 25(3):1160–1168.

Van Den Berg, J., Guy, S. J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer.

Van den Berg, J., Lin, M., and Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. In *ICRA*, pages 1928–1935. IEEE.

Vanegas, C. A., Garcia-Dorado, I., Aliaga, D. G., Benes, B., and Waddell, P. (2012). Inverse design of urban procedural models. *ACM Trans. Graph.*, 31(6).

Vaughan, C. L. and OMalley, M. J. (2005). Froude and the contribution of naval architecture to our understanding of bipedal locomotion. *Gait & posture*, 21(3):350–362.

Wang, H. (2015). A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph.*, 34(6):246:1–246:9.

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

Weiss, T., Litteneker, A., Jiang, C., and Terzopoulos, D. (2017a). Position-based multi-agent dynamics for real-time crowd simulation. In *Motion in Games*, MIG '17, pages 10:1–10:8, New York, NY, USA. ACM.

Weiss, T., Nakada, M., and Terzopoulos, D. (2017b). Automated layout synthesis and visualization from images of interior or exterior spaces. In *Comp. Vis. and Pat. Recognition workshop on Vis. Meets Cognition (CVPR)*, pages 41–47. IEEE.

Witkin, A., Fleischer, K., and Barr, A. (1987). Energy constraints on parameterized models. In *Proc. SIGGRAPH*, SIGGRAPH '87, pages 225–232, New York, NY, USA. ACM.

Witkin, A. and Kass, M. (1988). Spacetime constraints. *Proc. SIGGRAPH*, 22(4):159–168.

Wolinski, D., Lin, M. C., and Pettré, J. (2016). Warpdriver: context-aware probabilistic motion prediction for crowd simulation. *ACM Trans. Graph.*, 35(6):164.

Wu, W., Fan, L., Liu, L., and Wonka, P. (2018). Miqp-based layout design for building interiors. In *Comp. Graph. Forum.* Wiley Online Library.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In *Comp. Vis. and Pat. Recognition (CVPR)*, pages 3485–3492. IEEE.

Xie, S. and Tu, Z. (2015). Holistically-nested edge detection. In *Proc. of the Int. Conf. on Comp. Vis. (ICCV)*, pages 1395–1403. IEEE.

Xu, K., Ma, R., Zhang, H., Zhu, C., Shamir, A., Cohen-Or, D., and Huang, H. (2014). Organizing heterogeneous scene collections through contextual focal points. *ACM Trans. Graph.*, 33(4):35.

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):56.

Yu, F. and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In *Int. Conf. on Learning Representations*.

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

Yu, L.-F., Yeung, S.-K., and Terzopoulos, D. (2016). The clutterpalette: An interactive tool for detailing indoor scenes. *IEEE Trans. Vis. Comp. Graph.*, 22(2):1138–1148.

Yu, Q. and Terzopoulos, D. (2007). A decision network framework for the behavioral animation of virtual humans. In *Symp. Comp. Anim.*, SCA '07, pages 119–128.

Zhang, E., Cohen, M. F., and Curless, B. (2016). Emptying, refurnishing, and relighting indoor spaces. *ACM Trans. Graph.*, 35(6).

Zhang, Y., Pettré, J., Peng, Q., and Donikian, S. (2009). Data based steering of virtual human using a velocity-space approach. In *Motion in Games*, pages 170–181. Springer.

Zhu, L., Xu, W., Snyder, J., Liu, Y., Wang, G., and Guo, B. (2012). Motion-guided mechanical toy modeling. *ACM Trans. Graph.*, 31(6):127.

Zhu, Y., Jiang, C., Zhao, Y., Terzopoulos, D., and Zhu, S.-C. (2016). Inferring forces and learning human utilities from videos. In *IEEE Conf. Comp. Vis. Pat. Rec. (CVPR)*, pages 3823–3833.