DOI: 10.1111/cgf.13585

COMPUTER GRAPHICS *forum*
Volume 38 (2019), number 1 pp. 455–469

Urban Walkability Design Using Virtual Population Simulation

C. D. Tharindu Mathew[1], Paulo R. Knob[2], Soraia Raupp Musse[3] and Daniel G. Aliaga[1]

[1]Computer Science, Purdue University, USA
mathewc@purdue.edu, aliaga@cs.purdue.edu
[2]VHLAB, Pontificia Universidade Catolica do Rio Grande do Sul, Brazil
paulo.knob@acad.pucrs.br
[3]FACIN, PUCRS, Brazil
soraia.musse@pucrs.br

Abstract

We present a system to generate a procedural environment that produces a desired crowd behaviour. Instead of altering the behavioural parameters of the crowd itself, we automatically alter the environment to yield such desired crowd behaviour. This novel inverse approach is useful both to crowd simulation in virtual environments and to urban crowd planning applications. Our approach tightly integrates and extends a space discretization crowd simulator with inverse procedural modelling. We extend crowd simulation by goal exploration (i.e. agents are initially unaware of the goal locations), variable-appealing sign usage and several acceleration schemes. We use Markov chain Monte Carlo to quickly explore the solution space and yield interactive design. We have applied our method to a variety of virtual and real-world locations, yielding one order of magnitude faster crowd simulation performance over related methods and several fold improvement of crowd indicators.

Keywords: human simulation, animation

ACM CCS: Computing methodologies → Interactive simulation, Procedural animation

1. Introduction

Urban modelling is becoming increasingly popular in computer graphics and urban planning applications. One important aspect for urban spaces is designing and optimizing the environment for the walkability of pedestrians [Sou05, ADLN12]. Indeed, walkable cities have been linked with a healthier life, spending less time in traffic and having more time for recreational activities. This paper provides a tool where environment characteristics can be altered in order to provide a near-optimal configuration for a desired crowd behaviour.

Previously, researchers have investigated urban procedural modelling and crowd simulation. On the one hand, procedural modelling has become popular because it enables generating complex environments from a compact set of rules and parameters [PM01, WWSR03, VAW*09]. On the other hand, crowd simulation seeks recreating realistic crowds by modelling the behaviour of the individuals [PAKB17, vdBLM08, dLBRM*12, TM13, BPA16, KPAB15]. However, crowd simulation usually assumes that people know the location of their goals (or destinations). Also, the configuration of the environment is fixed and people generally do not react to elements in the environment, except to a few concepts such as density [Hug02, TCP06, BNCM14].

In contrast, our approach consists of three main components. First, we propose a parameterized procedural representation of a walkable environment, which includes:

- a network of generated virtual walkways, or even walkways obtained from a real-world location,
- walkway widths,
- a flexible representation for visual stimuli, i.e. signs and
- a set of goals.

Each sign present in the environment has a level of appeal (e.g. size, clarity, etc.), and information about a set of goals also located within the environment. Our concept of signs is kept general and refers to stimuli that influence an agent towards one or more goals.

Second, we extend an existing crowd simulation framework [dLBRM*12] to support an exploratory behaviour, where pedestrians

are able to explore the environment while not necessarily knowing the location of their goals; i.e. agents can reach their desired goals by chance or by getting stimulated by signs along the way to learn its location. The signs are readable by agents at a prescribed maximum distance and they affect them based on a function defining the interaction. As a consequence, agents can react (or not) to the sign's information. Further, the simulation supports changing the pedestrian walkway network and includes several acceleration schemes to yield fast performance.

Third, we provide an inverse procedural modelling framework, using an efficient large-scale optimization scheme, to explore the model design space and arrive at a solution having desired indicator values. Our indicators support urban walkability design concepts [MCG17, GJBP96] and include measures of time taken, distance walked, walkway cost and interpersonal distance [HAL63].

Succinctly, our main contributions include:

- a procedural model of urban space (based on realistic-world data or a virtual space) that is parameterized by sign locations and appeals, goal locations, walkway widths and walkway geometry (e.g. intersection locations, topology);
- a fast simulation tool for modelling how crowds explore unknown urban pedestrian areas that builds upon a space-discretization crowd simulator and uses urban signs and a contagion concept to model how people navigate to their destinations and (potentially) change their intentions and
- an optimization-based inverse-modelling engine that discovers an instance of the procedural model that yields a desired crowd behaviour.

We have used our approach to design and alter various real-world and virtual urban spaces. For existing locations, our tool provides what-if scenario exploration (e.g. seeking the smallest/lowest-cost changes that would improve the crowd flow as desired). For virtual urban spaces, our tool permits carte blanche design of novel spaces. Altogether, our environments range up to several hundred thousand square metres and include the simulation of several thousand people, yielding typical crowd behaviours. Our optimized designs are able to produce a several-fold decrease, or increase, in travel time or distance walked, for example. Moreover, using our interactive inverse design system, users can obtain a design in under a minute to a few minutes on a desktop PC using our acceleration schemes and a multi-core implementation.

2. Related Work

Our work builds upon procedural modelling and crowd simulation. On the one hand, urban procedural modelling has become extremely versatile and widespread [VAW*09]. The seminal work of Parish and Mueller [PM01] and subsequent improvements have addressed building design [WWSR03, MZWVG07], parcel design [LSWW11, VKW*12], road design [CEW*08, NGDA15], traffic design [SWL11] and recently weather design [GDAB*17]. Nevertheless, the Achilles' heel of procedural modelling is preemptively obtaining an urban configuration that yields a desired result. Towards this goal, inverse procedural modelling attempts to determine a procedure (e.g. the rule set, its application sequence or both) yielding a desired output. For example, Stava et al. [SBM*10] discovered procedural description of 2D vector geometry, Bokeloh et al. [BWS10] and Demir et al. [DAB15, DAB16] discovered symmetries and repetition to find a procedural description of a 3D urban model, Talton et al. [TLL*11] stochastically drive a procedural model so as to obtain an output following a desired global shape, Vanegas et al. [VGDA*12] altered the shape of a city to obtain desired urban indicator values, Garcia-Dorado et al. [GDGAVU14] changed the road network so as to obtain a desired vehicular traffic pattern over time and Nishida et al. [NGDA*16] inversely find a procedural grammar that yields the interactively drawn digital sketch of a building.

On the other hand, many different methods exist for crowd simulation [PAKB17, TM13]. Crowd models may be characterized as microscopic (agent-based) or macroscopic (aiming to capture statistical properties such as crowd density and flow at a global level). In an agent model, each individual perceives and reacts to the world according to various local rules [Rey87, PAB07, GCK*09], social forces [HFV00, TCP06, Che04], following behaviours [LJK*12] or velocity obstacles [vdBLM08, KGM13]. Macroscopic approaches aim to govern the global behaviour of crowds using environment descriptions [YMC*05, PGT08], space colonization [dLBRM*12] or continuum fluid-like flows [TCP06]. In general, in crowd simulation methods, people know where they want to go and explicitly know the goal locations besides other pre-defined information (e.g. initial positions and fixed obstacles positions). Typically, a person's changing intentions while exploring a space are not a focus in crowd simulation. However, some methods for crowd evacuation planning have agents that react to the environment [BUH*15, COMB16, HUB*15, HUB*16, BNCM14, HJ09].

Other recent work uses crowd simulation-based evaluation to interactively provide feedback to building designers (e.g. [HUB*17]). [LPH*17] uses neural networks to encode the environment design, crowd distribution and steering method to predict the feasibility of the layout design, thus avoiding running full simulations and yielding faster times. [CBH*17] uses crowdsourcing to help evaluate the design of building layouts based on a user feedback system. Varying crowd intentions, exploration assistance using signs and walkability criteria of city environments are not a primary focus of these recent work.

Some authors propose methods where agents perceive and react to signs in the virtual environment (e.g. traffic and exit signs). Helbing presented emergent behaviours and also discussed the effect of people reacting to the environment and traffic signs [HFV00, HJ09]. [TB16] presents a rule-based reasoning system to identify potential traffic sign locations. In [Tor15], the authors introduce a framework for addressing relationships between Geographic Information Systems (GIS), models of physical and human processes and immersive 3D worlds. Recently, in [NYF*15], authors discuss the influence of safety signs in evacuation processes, since the shared information impacts the agents' decision during evacuation. Bode [BKWC13] proposes that agents in the crowd choose between different exit routes under the influence of three different types of directional data: static information (signs), dynamic information (movement of simulated crowd) and memorized information, as well as the combined effect of these different sources of directional information. In a recent work, [HLB*17] approaches the automatic placement of

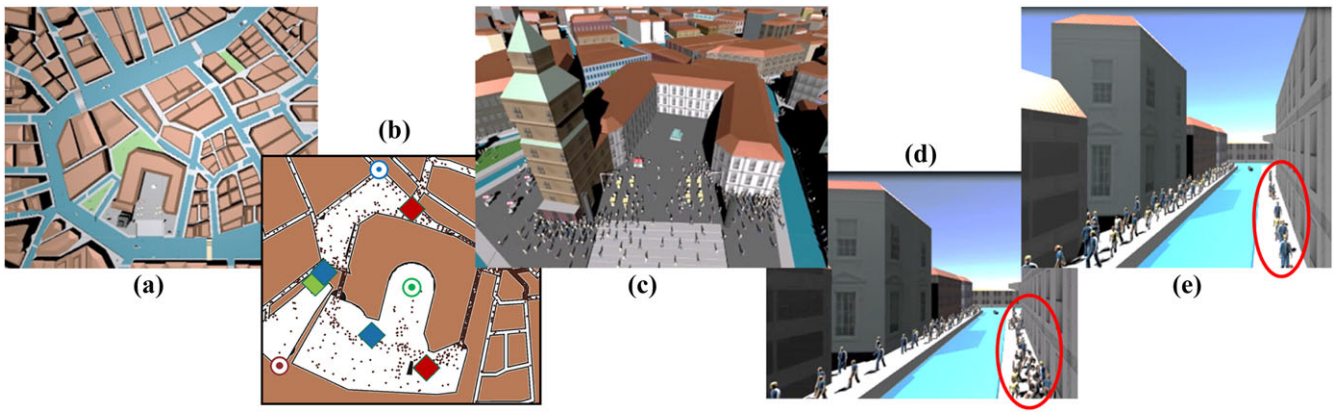

Figure 1: *Urban walkability design: Our system takes as input (a) an urban layout (e.g. Venice), a number of agents (e.g. 1000), intentions and goals in order to determine layout modifications that yield a desired crowd behaviour where agents are initially unaware of the goal locations ((b) top down 2D view and (c) a 3D view with virtual humans). The system enables altering crowd behaviour (e.g. higher (d) or lower (e) agent density) by automatically positioning signs of variable appeal, changing goal locations and/or altering walkway widths and topology.*

signs based on a two-step process. The first step finds an optimized path using way-finding optimization, which is then followed by a second sign placement step based on agent-based navigation.

In contrast to previous research, our goal is to discover a configuration, or least cost alteration, of an urban pedestrian layout for agents who are exploring the environment (i.e. it is initially unknown) and we wish to yield a desired crowd behaviour. The two most similar works are Garcia-Dorado et al. [GDGAVU14] and Feng et al. [FYY*16]. Garcia-Dorado et al. [GDGAVU14] aim to alter a road network and a configuration of traffic lights to yield a desired spatio-temporal vehicular traffic behaviour. In a similar manner, we seek a set of pedestrian walkways, signs and goal locations that yield a desired crowd behaviour. This enables passively controlled crowd behaviour and is beneficial to both urban planning and to design and simulation. Feng et al. [FYY*16] propose an approach to generate mid-scale layout designs optimized by crowd properties that assume each agent knows the locations of their destinations. They use non-linear trained regressors to avoid hundreds of long-time simulations and, given a domain like a shopping mall, the method can find the paths and sites optimizing mobility, accessibility and coziness. Even though Feng et al.'s work has several similarities with our own, we address a different scenario whereby agents do not know *a priori* their goal locations. Using the shopping mall domain, in our system, agents may enter the building with no knowledge about goal locations and will guide themselves using signs and according to their intentions. Our system optimizes the placement of signs and the topology of the walkways in order to allow agents to reach their goals in a faster way, as well to consider the cost of creating new walkways or adding signs.

Overall, our approach has three main differences as compared to prior work: (i) we use the contagion concept from [BDM*15] to model the interaction among people and urban signs and to provide a way of simulating the sharing of visible information in an initially unknown pedestrian environment; (ii) we use inverse modelling to determine a parameterized procedural representation of an urban walkway system and (iii) we simulate large-scale human crowds' one order of magnitude faster than [FYY*16] and [dLBRM*12].

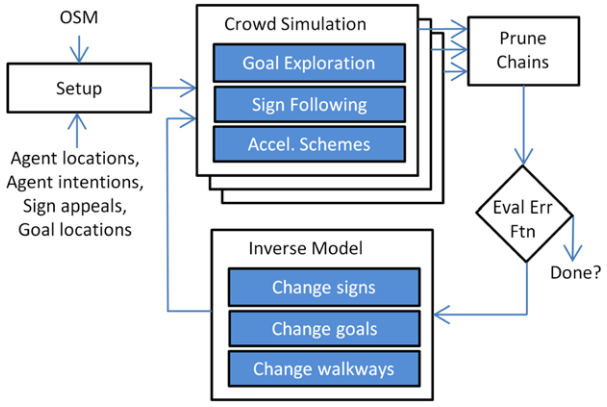

Figure 2: *System pipeline: Overview of our method including the data flow and processes to simulate and alter environments.*

3. Overview

This section presents our approach to generate urban layouts that produce a desired population behaviour. Figure 2 shows an overview of our computational pipeline. The remainder of this section details the urban procedural model and a summary of the user interaction with our design tool.

3.1. Urban procedural model

Our system includes an engine to generate an urban procedural model U consisting of a set of walkways of desired widths, signs, goals and start locations. The walkways can be procedurally generated or imported from OpenStreetMap (OSM). In both cases,

Figure 3: *Example urban layouts: (a) mostly pedestrian area of central Strasbourg, France, (b) Commonwealth Park in Canberra, Australia, (c) pedestrian area of downtown Zurich, Switzerland and (d) a snapshot of our GUI.*

the walkways are represented as graphs and we support operations such as add/remove corners and edges and generate walkway geometry in between corners. Figure 3(a) (a pedestrian part of central Strasbourg, France), Figure 3(b) (Commonwealth Park in Canberra, Australia) and Figure 3(c) (a mostly pedestrian area of downtown Zurich, Switzerland) contain example areas extracted from OSM. While highlighted that OSM notation is quite liberal and contains a large set of tags—we only process tags relevant to walkways and automatically extract them.

Once the walkway structure is established, our system allows to use an externally provided characterization of the virtual humans (i.e. their start locations, ordered set of goal labels and intentions and average speed), or we can algorithmically generate virtual humans and its said characterization. The urban layout will have several start locations (e.g. parking lots, bus stops) from which virtual humans will enter the environment and a set of goal locations. Each human will have a set of intentions with different values to visit one or more goals. Virtual humans commence at a small random offset from their associated start location. Further, initial locations for signs in the environment may explicitly specified or may be automatically generated by our system (further details in Section 5.3, Sign Locations and Appeal).

3.2. User interaction

Using our GUI (shown within Figure 3d), the user may iteratively enable different ways of altering the environment and may specify

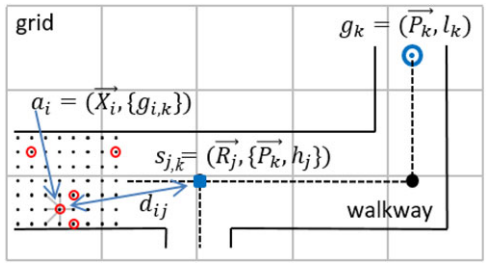

Figure 4: *Crowd simulation: We show the multiple variables used to simulate agents moving within a discretized space influenced by signs. The main entities are: an agent i (a_i), a goal k (g_k) and a sign j pointing to a goal* k *($s_{j,k}$).*

different crowd behaviour objectives by giving target values for our indicators: time taken, distance walked, walkway cost and number of people within intimate interpersonal distance. For example, the user may choose to fix walkway geometry and number of signs. Then, our inverse modelling engine automatically determines the best sign locations to accomplish a desired crowd behaviour (e.g. reduction in total time it takes all agents to visit all their goals). Alternatively, the user may wish to achieve a desired crowd behaviour by simultaneously changing the pedestrian walkways and using a fixed number of signs while maintaining a constant total amount of walkways. In Section 6, we present several different optimizations and walkway alterations. In all cases, after the optimization a 3D procedural model of walkways or an OSM-compatible geometry can be exported.

4. Crowd Simulation Model

To model the behaviour of our agents in unknown environments, we extend and accelerate a crowd simulation model based on space discretization, because it scales well and implicitly provides a guaranteed collision-free crowd simulation. Agents in the environment perceive a set of markers on the ground within their observational radius and move towards its immediate goal, following a resultant motion vector calculated as a function of its markers (i.e. see the markers surrounding agents i (a_i) in Figure 4). More detailed information is available in the work of Bicho [dLBRM*12]. This aspect of our system is based on the BioCrowds simulator [dLBRM*12] which supports some of the important emergent behaviours expected in crowd simulation (see the figures given in Section 6). In a space discretization method, such as ours, obstacles are very easy to represent as zones without any markers. However, we extend such a framework by introducing the notion of goal exploration and sign-following (with varying appeal), which influences agent movement. In addition, we add several enhancements that yield approximately one order of magnitude increase in simulation performance over [dLBRM*12]. These improvements are vital to our inverse modelling objectives.

4.1. Goal exploration

Firstly, a goal k is defined as $g_k = (\vec{P}_k, l_k)$, where P_k states for the location of goal k and l_k is the goal identifier. In our simulator,

agents have to discover the location of their intended goals. In other words, goal locations are not explicitly provided to each agent at the beginning of the simulation; instead, they are only provided with goal labels. This change in goal definition implies that since agents are not aware of goal locations, they have to explore the environment to find them. In addition, agents have a different *intention* associated with each of their goal labels; i.e. a number representing how much they want to go to each goal. More precisely, each agent i at position $\vec{X}_i$ has a list of n_i goals where each goal is defined as $g_{i,k} = \{l_k, \gamma_{i,k}\}$, for $k = 1, ..., n_i$ where l_k represents the id of goal k that the agent is interested to go to. $\gamma_{i,k}$ is the intention of agent i to go to goal k and has values in the range [0; 1] (Figure 4).

4.2. Sign following

We use *signs* to communicate the position of the goals in the urban layout to the virtual humans. The signs represented in this work simulate signs transferring knowledge of the global goal location to the virtual human, instead of simply pointing the agent in the correct direction. While a variety of signs, such as directional boards and elaborate maps, are possible, we focus on a single entity to represent all types of signs. The main purpose of signs is to introduce a mechanism that dynamically alters an agent's intention and current destination.

A sign j referring to a goal k (e.g. a place) is defined as $s_{j,k} = (\vec{R}_j, \{\vec{P}_k, h_j\})$, where $\vec{P}_k$ is the position of the goal k and h_j is the appeal of sign j positioned at location $\vec{R}_j$ (see Figure 4). The appeal h_j (in range [0; 1]) aims to represent a sign's attractiveness and visibility. We link the appeal value to a maximum observational distance of each sign—we assume that a sign is visible by at most $20 \times h_j$ metres away. For example, an appeal $h_j = 1.0$ could imply a strong auditory cue (which can be heard from 20 m away), $h_j = 0.5$ might represent a large billboard visible at 10 m and $h_j = 0.3$ potentially signifies a small walkway sign visible from 6 m or less. During simulation, agents can 'virtually see' nearby signs and obtain goal positions to guide their motion—conceptually the information they obtain can be a position on a map, GPS coordinates for smart phone usage, etc. An additional aspect of this definition is how much each sign impacts each agent in the crowd simulation. This is critical to our goal of wanting to simulate the dynamic behaviour of being influenced by the environment in order to change goals. For instance, an agent i that has an intention $\gamma_{i,k}$ to go to goal k can be influenced by the environment, i.e. after reading a sign j, the agent can change its intention to a lower or higher value.

Our sign interaction method is based on a contagion technique proposed by [BDM*15]. Their work is designed to model the contagion of one unspecified emotion in agents of a group. Mathematically, the authors define the emotion of an agent as a value q in the range [0; 1] that represents the intensity of an unspecified emotion at a given instant. According to Bosse's model, $q* = \omega_{S,R}$, where $\omega_{S,R} = h_S \alpha_{S,R} \delta_R$ is the contagion between sender S and receiver R of a certain emotion. In addition, h_S is the expressiveness (i.e. appeal) of S, $\alpha_{S,R}$ represents the strength of the contagion and δ_R is the susceptibility of R. We adapted this concept to define $\gamma_{i,k}$, i.e. the intention of agent i to go to goal k and how much this can be influenced by signs in the environment. Bosse's equations were adapted to our method in order to compute a new intention for agent i to go to goal k after having interacted with sign j is as follows:

$$\gamma_{i,k}* = \gamma_{i,k} + (\omega_{i,j}(1 - \gamma_{i,k})), \tag{1}$$

where $\gamma_{i,k}*$ is the new intention value of agent i to go to goal k, and $\omega_{i,j}$ is given by $\omega_{i,j} = h_j \alpha_{i,j}$, where we assume a constant susceptibility and $\alpha_{i,j}$ is defined as a function of the distance between agent i and sign j:

$$\alpha_{i,j} = \min\left(1, \frac{1}{d_{ij}}\right), \tag{2}$$

where d_{ij} is the Euclidean distance between agent i and sign j. In the original model [BDM*15], the information representing the contagion strength among two individuals is fixed. We altered the model so that contagion strength $\alpha_{i,j}$ between the agent i and sign j varies depending on their mutual distance.

4.3. Agent movement

During the simulation, agents attempt to satisfy their intention list. Initially, each agent i is endowed with a list of n_i goals k (fixed) and per-goal intention values $\gamma_{i,k}$ (can change during the simulation) in the range $[0; T)$. In experiments, we assign a starting intention based on $X \sim \mathcal{N}(0.5, 0.1^2)$. The intention values associated with goals for each agent are ordered from highest to lowest value. One innate behaviour, called *Looking_for*, is instantiated to each agent and aims to provide a random motion in the urban layout, while $\gamma_{i,k} < T$ for all k goals in the agent i list. The goal of this innate behaviour is to apply the exploratory behaviour, while looking for signs to be informed. When it happens, $\gamma_{i,k}$ can be updated according to Equation (1). If $\gamma_{i,k} \geq T$ for a specific goal k, then the agent goal-based motion is applied, causing agent i to move towards goal k. The navigation of the agent is handled by making use of the shortest path between the agent's position and desired destination. These are computed using Dijkstra's shortest path on the procedurally generated walkway graph. This resembles an efficient medial axis navigation graph approach, as opposed to a navigation mesh approach [vTTK*16, PF16]. The agent is given the shortest path from its current location to the goal location (a small random offset is added to each agent's goal location so that multiple agents do not intend to arrive at precisely the same goal location). Once a certain goal is reached, it is removed from agent list. If $\gamma_{i,k} \geq T$ for more than one place k, then the highest intention value is prioritized. In order to avoid computing intention changes among all agents and signs, we only consider intention changes for agents within the maximal observational distance of each sign. Further, to enable agents actually 'seeing' their goal when near it, we include, unbeknown to our inverse modelling optimization, a fixed sign placed at each goal location and pointing to it.

4.4. Acceleration

We perform several enhancements to significantly accelerate the performance of our simulator as compared to the times reported in several other space discretization methods, including [dLBRM*12]

and [FYY*16]. This acceleration is beneficial to the interactive inverse modelling solutions described in the next section.

- *Graph Representation.* Instead of representing the walkways with a set of 2D polygons, we describe them with a 2D spatial graph and associate width values with each edge (e.g. see the skeleton connected by vertices inside the walkway and the width values w in Figure 4). For OSM-imported walkways, this is a very natural representation. For curved walkways, the representation is a bit inefficient but overall beneficial in light of the following enhancement.
- *Cached Shortest Paths*: Since the walkways are discretized to a set of nodes, we can easily calculate and cache the shortest path routes from all nodes to all nodes. This results in significantly fewer shortest path computations than using the dense set of markers of the space discretization framework for shortest path computations. At usage time, however, random offsets less than half the width of the walkways are added to the nodes defining the shortest path. Thus, the path computed for each agent is not necessarily the true shortest one, but almost.
- *Grid Cells*: The graph is then used to create a grid of markers (i.e. the space discretization as depicted by the grid in Figure 4). However, agents only need to inspect markers in their grid cell or in the neighbouring cells. This makes the system depend mostly on the number of agents and not on the size of the simulated area (such will be shown in Section 6).

5. Inverse Model

Our inverse model engine seeks for changes to an initial urban layout U_0 that yields a desired crowd behaviour. We quantify the crowd behaviour by using an error function that considers several indicators inspired by urban walkability concepts [GJBP96]. [GJBP96] established five design criteria (the 5Cs) to enable walkability (i.e. pedestrian friendly urban areas): connectedness, convenience, comfort, conspicuity and conviviality. Our indicators implement necessary measures to yield a first-order quantification of the first four of the aforementioned design criteria–we leave conviviality (i.e. the quality of landscaping and architecture and the diversity of activities) to future work. To quantify connectedness, we measure the time taken for agents to visit all their destinations while ensuring that all destinations are at least accessible (e.g. connected) but offer the ability to decrease/increase the total amount of walkways. To measure convenience, we merge the notion of time taken to visit all destinations with walkway width which reduces agents' congestion in a walkway and at intersections. Regarding the comfort, we count the average number of people within the intimate interpersonal distance range [HAL63]. Finally, for conspicuity, we measure the number, location and appeal of signs in the urban area.

5.1. Error function

Altogether, our optimization error function consists of a weighted sum of the following indicators used to quantify the aforementioned criteria:

- *Time Taken T*. We count the number of time steps it takes to complete a simulation. In our system, each time step typically corresponds to 1 s of simulated time. The time is measured until a percentage of the agents visit all their intended goals. We typically use 97% thus accounting for some agents being content in not visiting all their goal locations.
- *Average Walking Distance W*. This value represents the average distance walked by each agent during the simulation.
- *Walkway Cost C*. This metric calculates the area used up by walkways divided by the area of the bounding box surrounding the entire urban layout. Effectively, it measures the cost of having walkways.
- *Intimate Distance Count D*. This metric calculates the average number of people within the intimate interpersonal distance range of each agent during the simulation period.

The fourth indicator is inspired by the concept of personal space, or proxemics, proposed by Edward Hall [HAL63, Hal66]. Personal space is the region surrounding a person. Most people value their personal space and feel discomfort, anger or anxiety when their personal space is populated. For instance, considering two individuals i and j and their Euclidean distance $d_{i,j}$, the intimate zone ($d_{i,j} <= 0.45$ m) is reserved for close friends and close family members. Another zone (personal space, $0.45\,\text{m} < d_{i,j} <= 1.2\,\text{m}$) is used for conversations with friends and in-group discussions. A further zone (social space, $1.2\,\text{m} < d_{i,j} <= 3.6\,\text{m}$) is reserved for strangers or newly formed groups, while a fourth zone (public space, $3.6\,\text{m} < d_{i,j} <= 7.6\,\text{m}$) is used for speeches and larger audiences. Overall, the best value for this indicator is typically achieved if few or no people are in the intimate zone. The combined error function can be expressed as a linear combination $E = k_t T + k_w W + k_c C + k_d D$, where k_t, k_w, k_c and k_d are constants that enable giving different relative importances to the metrics. Normalization constants are embedded in T, W, C and D.

5.2. Optimization

Given an initial urban model U_0 and the aforementioned error function, we use a Markov chain Monte Carlo (MCMC)-based method [GRS95] and the Metropolis–Hasting algorithm [MRR*53, Has70] (with an acceptance probability of 0.9) to find a new set of walkways, signs and goals that minimize the error function (e.g. $E(U_i)$ -> 0). This optimization consists in simultaneously running many sets of Markov chains over a large number of iterations and choosing the best solution states. In particular, the optimization begins by seeding s initial configurations similar to U_0 and then using parallel tempering [SW86] to attempt n state changes at each of t different temperatures. Each of the mentioned similar initial configurations is obtained by randomly performing one state change as described in Section 5.3. Further, for each state change, an entire instance of the aforementioned crowd simulation is executed prior to evaluating the error function. In the end, the walkway configuration with the smallest error is chosen as the solution U^*.

The acceptance probability v of a move from a current state U_i to a new state U_{i+1} is given by the Metropolis ratio. Hence, the new state $U_{i+1} = U_i'$ with probability v and $U_{i+1} = U_i$ with probability $1 - v$. To improve optimization performance, we periodically prune the explored solution chains. In particular, every $m < n$ state changes, the chains are pruned by keeping a percentage of the top performing chains (e.g. top 20%) and re-initializing the rest (e.g. bottom 80%)

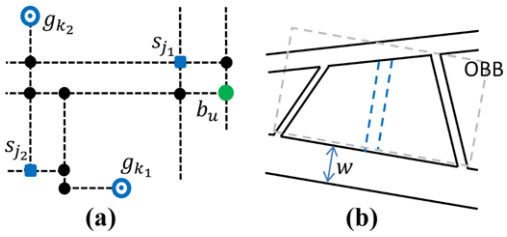

Figure 5: *Inverse modelling: We show possible layout changes during inverse modelling. (a) Given agent source locations, such as b_u, signs s_j* and/or goals g_k* may change positions to that of unoccupied nodes (i.e. the black nodes). (b) In addition, walkway widths may be altered or the oriented bounding box of a closed loop of walkways may be bisected to add new walkways.*

with small variants of the top set. In this way, the exploration focuses on the top solutions which most likely will lead to further improved solutions.

For all optimization scenarios, the constants have the default values $k_t = k_w = k_c = k_d = 1.0$, unless otherwise specified. We do not vary $k_t = 1.0$ and use this as a scaling factor to adjust other constants. For the connectedness criteria (Section 6.2.1), we increase $k_c = 2.0$ to discourage more additions of roads than necessary. For the conspicuity criteria (Section 6.2.2), all constants remain at 1.0. For comfort (Section 6.2.3), we increase $k_d = 2.0$ to encourage road thickness increase to increase interpersonal distance, and we increase $k_c = 2.0$ to control the increase of road thickness. For the convenience criteria (Section 6.2.4), we increase $k_d = 2.0$.

Moreover, we implement a job queuing system so that each chain is run on a separate core, thus yielding a parallel MCMC implementation. Typically, our solutions are run using five temperatures and 10 chains per temperature, pruning every 10th iteration and running a total of 20–50 iterations. This implies a total of several thousand crowd simulation executions, each simulation typically being a few thousand time steps (or one half to a few hours of simulated time). Our design system completes in under a minute to a few minutes, depending on the task, using a four-core desktop PC.

5.3. State changes

During the MCMC-based optimization, one of several types of state changes can be applied (Figure 5). The user can choose, via the GUI, which of the following state change types are enabled. Note that if more than one state change type is enabled, the system randomly selects one of the state change types to perform next.

- *Sign Locations and Appeal.* While the number of signs in the layout is specified by the user, the optimization determines their optimal location. An initial set of signs can optionally be provided in the initial urban model U_0. If not specified, then sign positions are randomly seeded at the beginning of the optimization and are assigned in a round-robin fashion for all goals (e.g. eight signs for four goals are set to be two signs per goal randomly positioned). All sign positions are assumed to be at graph nodes (e.g. at corners or at internal nodes of the edges when curved walkways are used). This results in the reduction of search space complexity for the optimization, at the cost of introducing a limitation—in the real world, signs can appear in the middle of a long walkway. Further, we do not allow for signs to be co-located with starting locations, goal locations (note that the optimization is unaware of the default sign associated with a goal as noted in Section 4.3) or with other signs having the same goal (i.e. signs for different goals can be co-located). During a potential sign-location state change, one sign at random is repositioned to a random new location. Sign changes do not make use of the multiple MCMC temperature settings. Sign appeal is global and all signs share this value. For the optimization, the sign appeal can be included as a cost. So, a lower appeal value minimizes the score. However, a lower appeal increases other indicators, such as time taken. For example, sign appeal will be reduced until the time taken increases too much.
- *Goal Locations.* This state change attempts to find the optimal locations for all goals. All goals have an initial location specified in U_0 and, similar to sign locations, can only be re-located to graph nodes (and may not be co-located with signs or other goals). During a potential goal location state change, the goal is randomly repositioned by a distance linearly proportional to the chain's temperature (e.g. a chain at temperature 2 seeks for a new goal position at up to twice the distance from its current location as compared to a goal move for a chain at temperature 1).
- *Walkway Width.* This state change type alters the width of a segment of walkway located between a pair of walkway intersection nodes (i.e. nodes internal to an edge are ignored). For a potential walkway width state change, the optimizer follows a random selection method to select an edge and to choose whether to perform a walkway width increase or decrease. The walkway width increment (or decrement) is proportional to the chain's temperature setting. The random selection process chooses with 80% probability a segment within the top 20% of segments sorted by walkway utilization (e.g. average number of agents on the segment during a prior simulation run).
- *Walkway Topology.* Our last state change type alters the topology of the walkways by either adding or removing a walkway segment. For a potential walkway topology change, the system randomly chooses to add or remove a random walkway segment. For segment removal, the system selects, with 80% probability, an edge from the last 20% of segments sorted by utilization. Removing a walkway segment is simply deleting an edge from the walkway graph. To add a walkway segment, first our system determines all loops (or cycles) in the walkway graph. Then, one of the loops is picked for walkway addition, by selecting with 80% probability, an edge from the top 20% of edges. (The walkway edge segments are sorted by the number of virtual humans utilizing the walkway). As the edge can be part of two loops, the system randomly picks one. Then, the oriented bounding box (OBB) of this selected loop is calculated, such that the area of the OBB is maximized (note that the OBB will be aligned to one of the edges in the loop). A walkway segment perpendicular to the OBB's major axis is then added to the walkway graph. Its width is the average of the width of the walkways it intersects. We experimented with using temperatures to add or remove walkway segments of size related to the temperature setting but did not find a significant performance improvement.

Table 1: *Simulation performance: We show the crowd simulation performance for different numbers of agents and in different sized environments.*

No. of agents	Environment area	Environment name	FPS
1000	1274 × 580 m^2	C.W. Park	286
5000			45
1000	480 × 340 m^2	Strasbourg	500
5000			54
1000	580 × 503 m^2	Venice	285
5000			50
1000	374 × 632 m^2	Zurich	333
5000			45
3000	677×514 m^2	Average performance in the above environments	200

An additional feature supported by our method is specifying a region of interest for the state changes. By default, the state changes may occur anywhere in the urban layout. However, the user may specify a rectangular area of the layout in which selected state changes may occur to further reduce computational cost.

6. Results

Our system is implemented on an Intel i7 desktop PC. The simulation and inverse engine are programmed in C/C++ using Windows, Qt, OpenGL and Boost (for graph operations). We implemented and applied our approach to several virtual urban layouts and real-world urban layouts; in particular, the three real-world environments in Figure 3 and a depiction of central Venice, Italy (Figure 1). For each of these locations, we inspected OSM to determine a set of plausible agent start locations and goal locations for typical people/agents. In all cases, when we report simulation results, we average five runs and present the average values. We have adopted this strategy because there is a random component in the simulation that can cause some variance.

6.1. Simulation performance

Table 1 reports the average number of crowd simulation time steps executed per second (i.e. FPS) for different numbers of agents and different environments. Typically, simulations run from 2000 to 8000 time steps (representing 2000–8000 s of simulated time). As can be observed, when a sufficient number of agents are reached, the performance is dominated by the number of agents (and independent of the environment size). The last row shows the overall average performance (3000 agents at 200 fps). For comparison, a simulation of 1000 agents in Feng et al. [FYY*16] runs at 24–60 FPS (when not using their trained statistical approximation) and Bicho et al. [dLBRM*12] run at 23 FPS (extrapolating performance for their numbers at 800 agents). Our system runs at an averaged 351 FPS for the same number of agents in our example environments (about one order of magnitude faster performance).

6.2. Walkability criteria

We show the results of altering the walkways for different walkability criteria. Typically, during solution computation, we initially compute a solution with 100–200 agents (e.g. for our large Common Wealth Park example, we obtain a simulation performance of about 3300 fps) to obtain a coarse solution and then run a few iterations with a larger number of agents. The total inverse modelling computation time for 100–200 agents typically is from under a minute to a few minutes. Refinements using a larger number of agents sum up to 10 min in few cases. The values for k_t, k_w, k_c and k_d are set as described in Section 5.2.

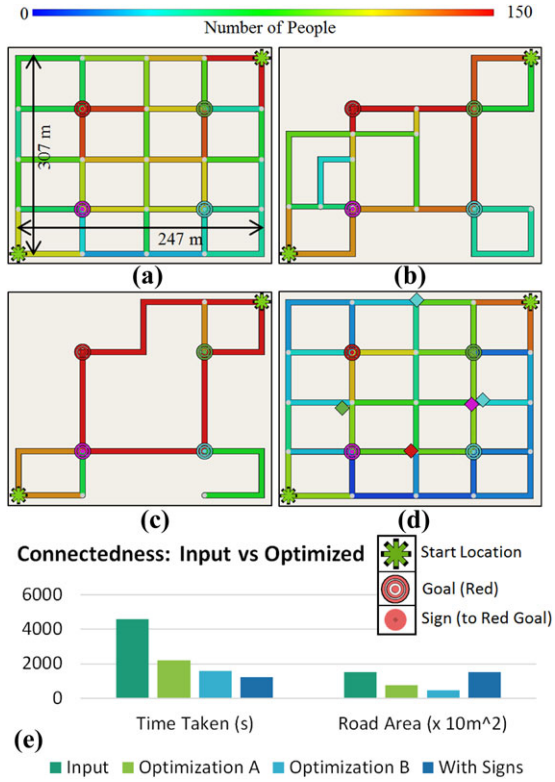

Figure 6: *Connectedness: 200 agents emerge from bottom-left and top-right green source with intentions of reaching four circular goals in middle. Walkways are colour coded based on the number of people who traverse each segment: (a) initial simulation result, (b) optimized with $k_c = 1$, (c) optimized with $k_c = 3$ and (d) alternative solution maintaining initial walkways but with five optimized signs.*

6.2.1. *Connectedness*

Figure 6 shows changes to the connectedness criteria. Using the same initial layout in Figure 6(a) with area 247 × 307 m, we use our system to discover alternative walkway topologies ensuring at least the two sources and four goals remain singly connected. In Figure 6(b), we choose $k_c = 1$ which converges to a walkway reduction of about 50% as well as less time taken for all agents to visit their goals. Figure 6(c) contains another solution where $k_c = 3$ resulting in about a 75% less walkways. As can be observed, reducing the number of walkways actually improves performance. This result is not unexpected and is an empirical demonstration of Braess' paradox (by giving agents less freedom, an overall win

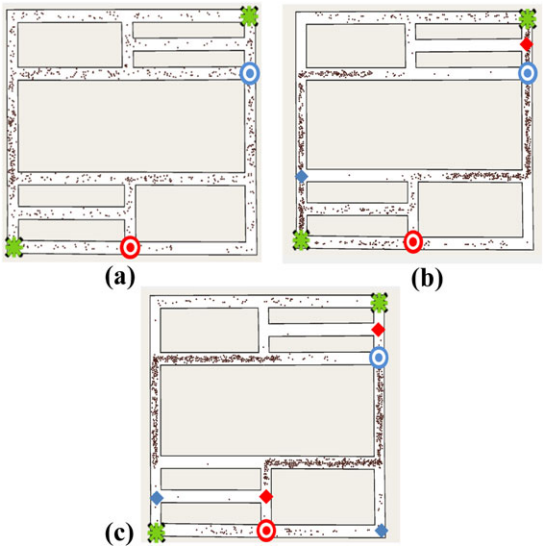

Figure 7: *Conspicuity in a virtual layout. (a) Environment with 1000 agents starting from two locations (bottom left and top right green dots), two goals and no signs. (b) Same environment but two optimized signs requires 5626 time steps for all agents to reach their single goal. (c) Same environment with four optimized signs reduces time taken to 3505 time steps.*

is produced despite there being less space and potentially higher congestion for the agents). Figure 6(d) shows a different solution whereby the original walkways are used but signs are added so as to yield a similar amount of time needed as with removing walkways. This shows our system's ability to enable alternative solutions within the same environment.

6.2.2. *Conspicuity*

Figure 7, Figure 8 and Table 2 show alterations to the conspicuity criteria. In effect, we alter the usage and placement of signs. In Figure 7, we construct a virtual layout to illustrate sign usage with 1000 agents. In all cases, the agents starting from the lower left wish to reach the goal in the upper right and the agents starting from the upper right wish to reach the goal in the lower right. In Figure 7(a), we show the base case of placing no signs, resulting in greater than 10 000 time steps necessary for 97% of the crowd to reach their single goal. Each dot corresponds to an agent at about midpoint through the simulation. In Figure 7(b), our inverse model determined the best location for two signs which reduced the time taken to 5626 time steps (note the proximity, as expected, of each sign to the corresponding agent source location). Finally, in Figure 7(c), our system determined the best locations to position four signs resulting in only needing 3505 time steps. It also becomes clear which are the most utilized walkways. Figure 8 shows a similar analysis for Common Wealth park in Canberra. We report the number of time steps for 10, 40 and 60 signs using an appeal value of 0.5. Further, we code the sign utilization using a standard Jet colourmap (i.e. red = 'high utilization'). At 60 signs, it can be seen that many signs have low usage. At 10 signs, they are almost all heavily used. Thus, 40 signs seem to closer to optimal in terms of reducing time steps and not having an excessive number of signs. Figure 8(d) shows the result of reducing the sign appeal value to 0.3 (i.e. making the signs less influential on people) and still maintaining a similar solution to the original 40 signs case. Thus, the end result has just enough conspicuity to achieve the shown reduction in time needed for all agents to achieve their goals.

Table 2 reports in tabular form the number of time steps taken for a crowd of 3000 agents to visit all goals for different configurations of sign appeal and agent intentions. In particular, we choose the Venice environment (see Figure 1) with nine fixed signs and three goals. We vary the sign appeal levels from low (0.05), to medium (0.1) and to a high value (1.0). Then, we randomly vary the agents intentions to have a dominant intention for up to one goal, two goals or three goals. Altogether this produces nine different scenarios. As expected, the scenario with low appeal and one strongly preferred goal per agent (though not necessarily the same goal) takes the most time for all agents to satisfy their objectives. In contrast, when having signs of high appeal and all agents have a strong intention to reach all their goals results in the least time for all agents to reach their goals. Thus, by altering appeal and intention levels, we can represent a variety of crowd types and still use our inverse modelling system to find walkway geometries, goal locations and sign locations that best accomplish our desired overall behaviour.

6.2.3. *Comfort*

Figure 9 describes changes to the comfort criteria, in particular to the average quantity of people in the intimate distance range. Figure 9(a) shows an initial configuration resulting in about 0.5–1 people within the intimate interpersonal distance (0.45 m) of the inset area. Figure 9(b) contains an optimized result changing walkway widths by the smallest amount to reduce the number of people in the intimate interpersonal distance, for the same inset area, to about 0–0.25 in most parts. While we observe average count values as high as 5, the most common and interesting values are near and under 1, so we artificially clamp counts to 1 prior to visualization.

6.2.4. *Convenience*

Figure 10 shows changes to an environment so as to alter the convenience of the walkways in allowing agents to quickly reach all their goals for a fixed topology and sign setup. In this case, only the walkway width is optimized. In this figure, 1000 agents enter the environment from bottom-left or top-right corner and each agent wishes to reach each of the four goals. Figure 10(b) shows a reduction in the time taken (as a consequence of less congestion) by using wider paths as determined by our optimization. For comparison, Figure 10(c) shows a naively assumed walkway width increase (equal to that determined by our optimization but for another subset of walkways) that should be beneficial but instead it shows a degradation of conditions. This is due to the virtual crowds reaching the thicker, wide areas and getting congested when trying to move out of them to the thinner walkways (creating a bottleneck). We experimented with slightly reducing the expanded walkway width to decrease chance of congestion but it only improved the performance

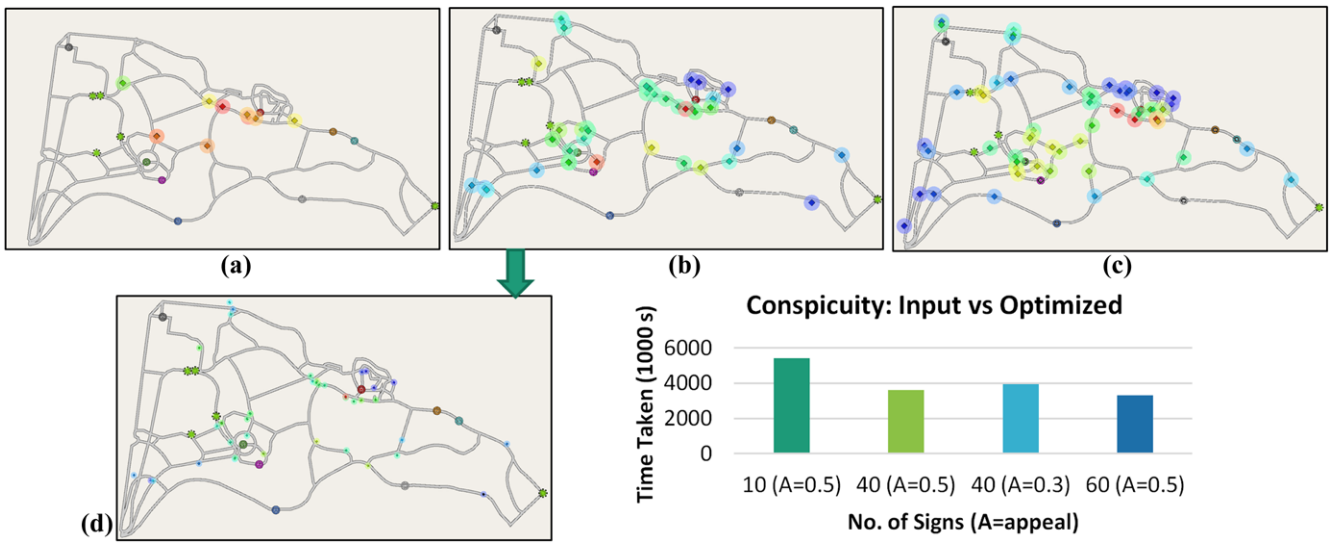

Figure 8: *Conspicuity in common wealth Park. Two hundred agents were initialized to go to eight goals from six locations in Common Wealth Park: (a) 10 signs, (b) 40 signs and (c) 60 signs. Sign dots are colour-coded by their utilization (the percentage of the virtual population that saw the sign) using a standard Jet colourmap (10 signs solution shows lots of red, i.e. most of the population saw the red signs and 60 signs show lots of blue, i.e. a small percentage of the population saw most signs). (d) 40 signs optimized to lowest possible appeal (from 0.5 to 0.3) yet maintaining the same time taken. Also, the circle around each sign shows its area of effect (thus circles in 'd' are smaller than in 'b' and 'c').*

Table 2: *Appeals and intentions: We show the number of time steps needed for the crowd to satisfy their objectives for various levels of signs appeal and agents' intentions. This demonstrates the ability of our system to capture different crowd behaviours and to use it within our inverse modelling framework.*

Appeal level	1 dominant	2 dominant	3 dominant
0.05 (low)	42 900	23 000	10 100
0.1 (medium)	24 800	13 500	8100
1.0 (high)	6900	6000	5700

slightly. Hence, this exemplifies the need for a thorough analysis before road changes are considered.

6.3. Real-world approximations

We used our system for several real-world approximate scenarios (Figure 13). The initial time is >10 000 time steps for all scenarios. In the left column, iterations of an optimization is shown for adding and removing, walkways and signs for Commonwealth Park. The optimization eventually finds an alternative set of walkways that reduces time taken. Note the non-axis aligned walkways added in the bottom image of the first column (iteration #60 of CW Park) in Figure 13. This solution shows the generality of our OBB-based algorithm (as described in Section 5.3, Walkway Topology) to not require axis-aligned walkways. In the middle column, we alter walkway width and signs for our Strasbourg area. We show segment

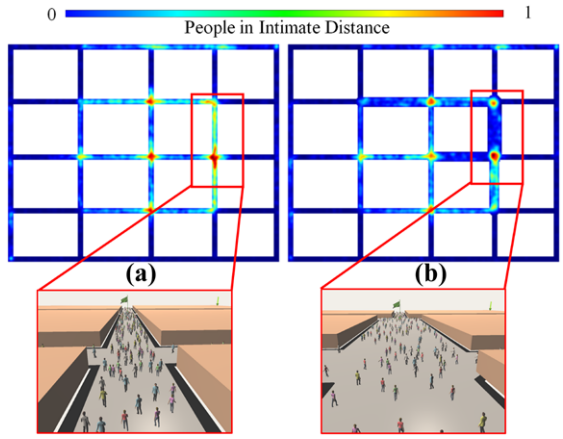

Figure 9: *Comfort: 2000 agents are initialized at bottom left and top right, with intentions of reaching four goals near middle. Each walkway segment is colour-coded to represent the number of people within every agent's intimate distance range (0.45 m). (a) Initial result and a zoomed 3D inset. (b) Walkway optimized to reduce number of people in intimate distance range by altering walkway widths, and a zoomed 3D inset.*

utilization at 20 iterations and sign utilization at 40 iterations. In the right column, we compute alternative goal locations and signs that reduce time taken in our Zurich environment. Collectively, these examples show how our system can be used in variety of real-world scenarios and for various optimization objectives.

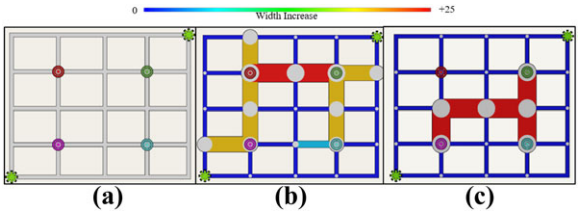

Figure 10: *Convenience: (a) 1000 agents were initialized at bottom left and top right, with intentions of reaching four goals near middle. This layout requires 4500 s for all agents to reach their goals. (b) Optimized with walkway width changes to reduce time by 3650 s but not exceed a prescribed total walkway cost. (c) As a comparison, naive walkway width changes which increases time taken to 4760, indicating that achieving the improvement is not trivial.*

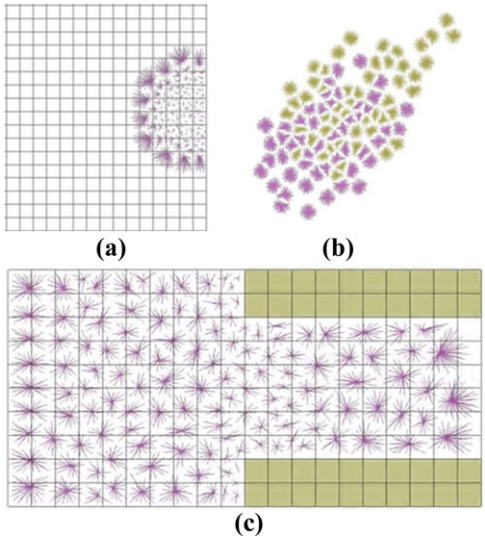

Figure 11: *Crowd simulation. We show emergent phenomena being produced by our crowd simulator in a manner similar to other crowd frameworks. Each star represents a virtual human. Each point of the star represents a marker that is acquired by the virtual human. The phenomena include: (a) arc formation, (b) emergent lanes (green and purple agents start at opposite corners and move towards the start position of the other group) and (c) bottleneck behaviour (all agents move from left to right). In (c), the agents squeezed to the upper and lower corners in trying to enter the walled area show the formation of a bottleneck.*

6.4. Evaluation and comparison

To evaluate our crowd simulator, we demonstrate the occurrence of emergent phenomena typically desired in crowd simulation (Figure 11) and compare our system to prior crowd simulators and exploration methods (Figure 12). In particular, as shown in Figure 11, we observe naturally occurring arc formation, emergent lanes and bottlenecks similar to Figures 6(b), 3(b) and 6(a) in [dLBRM*12], respectively.

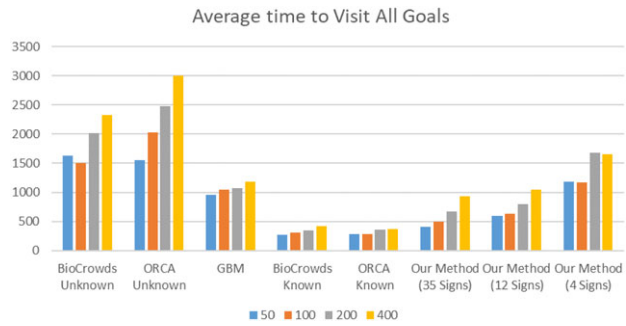

Figure 12: *Comparison. We compare our method to BioCrowds [dLBRM*12], ORCA [vdBLM08] and a graph-based method (GBM) having some randomization and agent memory. We show times for both known and unknown goal locations in the environment. Our method (right three cases) shows that we can be similar with BioCrowds/ORCA times for known goal locations despite assuming initially unknown goal locations and map. When we compare with a GBM that utilizes randomization and memory (when visiting multiple goals), we see that as we increase signs in our method, it still performs better. We varied the number of signs used in our method (4, 12 and 35) to show the impact in total time to achieve goals.*

In Figure 12, we compare our system to others. This comparison is to provide a benchmark for when agents are unaware of goal locations but the environment has signs and other design improvements. We compare with three different situations: known goal locations, unknown goal locations using graph search and memory to progressively explore the map and random movement. As expected, when agents know goal locations, the overall time needed to reach all goals is significantly lower than when goal locations are not known. However, with our system, we can design an environment that uses signs and our other design improvements to yield goal finding time that is similar to when all goal locations are known *a priori*.

Our comparison makes uses of several configurations of BioCrowds [dLBRM*12], ORCA [vdBLM08] and a graph-based method (GBM). We obtained implementations of BioCrowds and ORCA from the corresponding authors. In the space of autonomous agent navigation, many classical (graph search, potential fields, cell, etc.) and heuristic methods (neural networks, genetic algorithms, etc.) exist to explore a space [MCTK16]. To provide a basic comparison to these methodologies, we implemented a GBM by extending depth first search. At each intersection (i.e. node in the graph to explore), the visit order of the adjacent walkways (i.e. edges) is randomized. In addition, each agent remembers goal locations it passes by. Then, if an agent wishes to visit a previously seen goal, it simply takes the shortest path to this goal.

All the aforementioned systems are deployed in the environment shown in Figure 6. Further, each agent desires to reach each of four goals in a random order and moves at a speed of 1.2 m/s. We use the systems (i.e. BioCrowds, ORCA, GBM, our method) in simulation runs with 50, 100, 200 and 400 agents and in the three scenarios explained below.

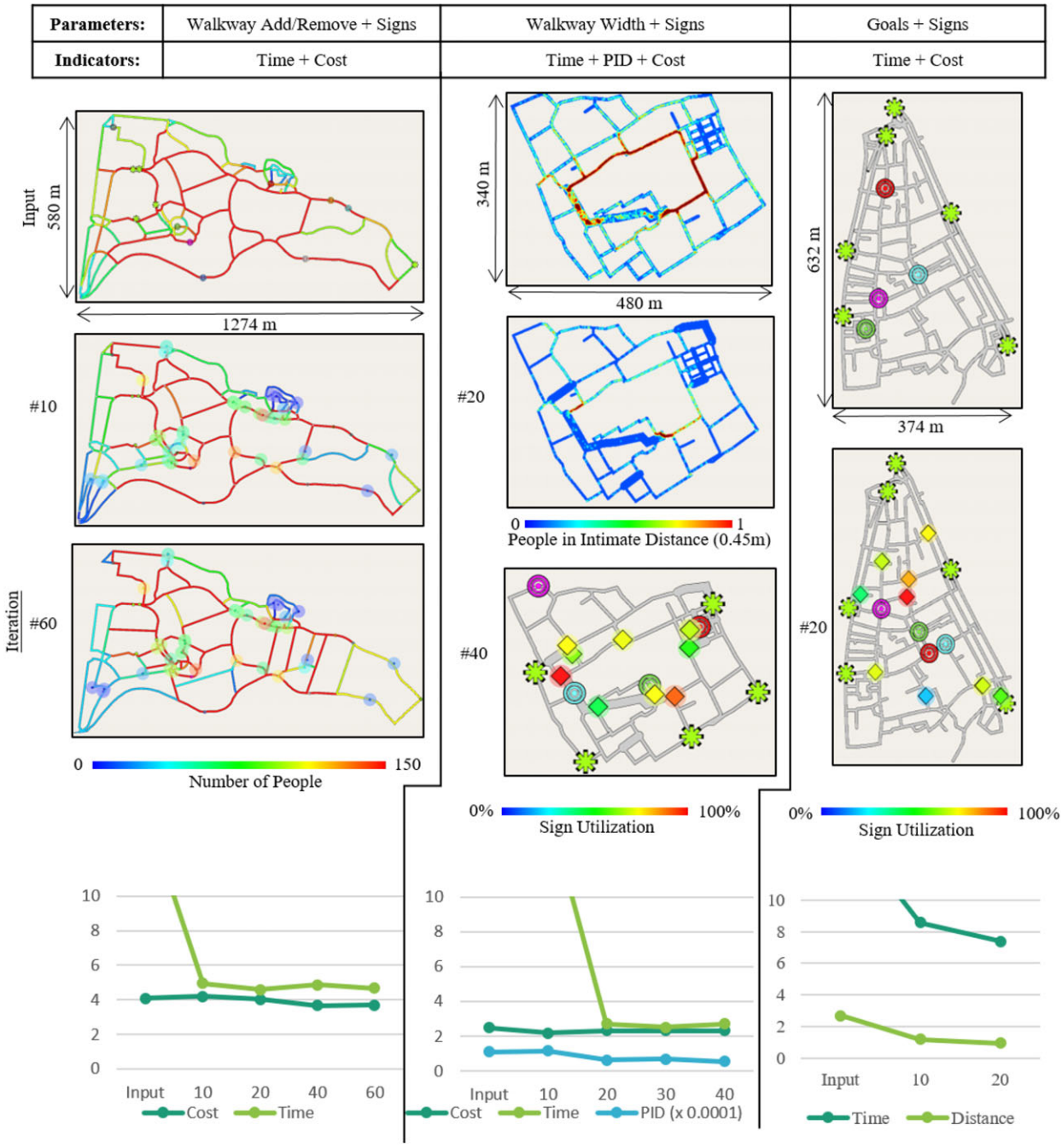

Figure 13: *Example applications. These three sequences (CW Park with 400 agents, Strousberg with 1200 agents and Zurich with 2000 agents) show how our system can be used in variety of real-world scenarios and for various optimization objectives ('time' refers to time taken, 'cost' refers to walkway area and 'PID' to the number of people within the intimate distance). See main text for more details.*

In a first scenario, we assume that agents know the location of all goals, as well as having a full map of the environment. This enables agents to compute the shortest path to all goals—this is the typical scenario for BioCrowds and ORCA. The time needed varies from 270 to 419 s. These are the times we seek to match but without the assumption that all agents know the location of all goals as well as the map.

In a second scenario, we assume that agents from BioCrowds and ORCA do not know the location of the goals, i.e. they move randomly until they achieve their goals. In this case, both methods show a high total time needed for all agents to visit their goals (from 1497 to 3002 s). We included this second scenario to have a reference point for when agents are unaware of goal locations and in addition, the environment is not informed with signs nor has our

other improvements. Our method should never be worse than this scenario.

In the third scenario, we use our GBM which essentially provides a middle ground solution. As agents explore and see goals, they effectively learn goal locations and the map. In this case, agents find all their goal locations faster (963–1187 s) than random search (1497–3002 s) but slower than when all goals are known *a priori* (270–419 s).

In contrast, our system is precisely for assisting, and modelling, agents exploring unknown environments. In this test environment, our system determines that with 12 or more signs, we obtain only a slightly longer time for all agents to reach their goals as compared to BioCrowds and ORCA using known goal locations (574–1056 s) and shorter times as compared to GBM (963–1187 s). With only four signs, our system predicts that it will take 960–1619 s, which is still less than the times predicted for the unknown environment as per BioCrowds and ORCA but more than with our GBM. These results show a progressive assistance by our method as the number of signs increases (in addition to our other improvements) and thus at least partially corroborates the correctness of our solution.

6.5. Limitations

Our system does have some limitations in its current state. First, we do not support external influences that may alter crowd behaviour (e.g. weather, emergencies or scenario-specific special information). Second, our framework is currently not well suited to accommodate long-term simulations which may exhibit complex agent behaviours (e.g. day long schedules, clustering of agents into groups/families or a cluster of agents getting delayed or faster based on the actions of a single group member). Third, our approach is heuristic-driven and thus may not find the globally best solution. Fourth, our system does not account for agent–agent interactions (beyond collision avoidance).

7. Conclusion and Future Work

We have described an interactive system that enables a user to inspect modifications to a set of walkways so as to alter the behaviour of a crowd exploring an unknown environment. We anticipate that our tool is useful for the animation of large crowds, whereby a simple set of rules is desired to simulate a large crowd but yet certain realistic global behaviours are desired. Indeed, our tool is also of interest to the urban planners we have contacted as an initial exploratory method to design, or alter, an urban layout for improved walkability. Our accelerated simulator and multi-core MCMC optimization makes the tool fast enough to provide an on-the-spot interactive analysis of the complex solution space.

As future work, we are interested in pursuing several items. First, our framework only has a small and limited set of crowd behaviour indicators and we would like to extend such indicators. Second, our system is limited to single-floor environments and thus we would like to extend to support multi-floor (indoor) locations. Third, agents within a crowd may behave differently based on background, culture and social function. It would be an interesting exploration to investigate how to alter an environment for such a diverse crowd of agents.

References

[ADLN12] Adkins A., Dill J., Luhr G., Neal M.: Unpacking walkability: Testing the influence of urban design features on perceptions of walking environment attractiveness. *Journal of Urban Design 17*, 4 (2012), 499–510.

[BDM*15] Bosse T., Duell R., Memon Z. A., Treur J., van der Wal C. N.: Agent-based modeling of emotion contagion in groups. *Cognitive Computation 7*, 1 (2015), 111–136.

[BKWC13] Bode N. W. F., Kemloh Wagoum A. U., Codling E. A.: Human responses to multiple sources of directional information in virtual crowd evacuations. *Journal of The Royal Society Interface 11*, 91 (2013). https://doi.org/10.1098/rsif.2013.0904

[BNCM14] Best A., Narang S., Curtis S., Manocha D.: Densesense: Interactive crowd simulation using density-dependent filters. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2014), Eurographics Association, pp. 97–102.

[BPA16] Beacco A., Pelechano N., Andújar C.: A survey of real-time crowd rendering. In *Computer Graphics Forum* (2016), Vol. 35, Wiley Online Library, pp. 32–50.

[BUH*15] Berseth G., Usman M., Haworth B., Kapadia M., Faloutsos P.: Environment optimization for crowd evacuation. *Computer Animation and Virtual Worlds 26*, 3–4 (2015), 377–386.

[BWS10] Bokeloh M., Wand M., Seidel H.-P.: A connection between partial symmetry and inverse procedural modeling. *ACM Transactions on Graphics 29*, 4 (July2010), 104:1–104:10.

[CBH*17] Chakraborty N., Berseth G., Haworth B., Faloutsos P., Usman M., Kapadia M.: Crowd sourced co-design of floor plans using simulation guided games. In *MIG '17: Proceedings of the 10th International Conference on Motion in Games* (New York, NY, USA, 2017), ACM, pp. 1:1–1:5.

[CEW*08] Chen G., Esch G., Wonka P., Mueller P., Zhang E.: Interactive procedural street modeling. *ACM Transactions on Graphics 27*, 3 (2008), 1–10.

[Che04] Chenney S.: Flow tiles. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), pp. 233–242.

[COMB16] Cassol V., Oliveira J., Musse S. R., Badler N.: Analyzing egress accuracy through the study of virtual and real crowds. In *Proceedings of 2016 IEEE Virtual Humans and Crowds for Immersive Environments (VHCIE)* (March 2016), pp. 1–6.

[DAB15] Demir I., Aliaga D. G., Benes B.: Coupled segmentation and similarity detection for architectural models. *ACM Transactions on Graphics 34*, 4 (July 2015), 104:1–104:11.

[DAB16] Demir I., Aliaga D. G., Benes B.: Proceduralization for editing 3d architectural models. In *Proceedings of 2016 Fourth International Conference on 3D Vision (3DV)* (October 2016), pp. 194–202.

[dLBRM*12] de Lima Bicho A., Rodrigues R. A., Musse S. R., Jung C. R., Paravisi M., Magalhes L. P.: Simulating crowds based on a space colonization algorithm. *Computers & Graphics 36*, 2 (April 2012), 70–79.

[FYY*16] Feng T., Yu L., Yeung S., Yin K., Zhou K.: Crowd-driven mid-scale layout design. *ACM Transactions on Graphics 35*, 4 (July 2016), 132:1–132:14.

[GCK*09] Guy S. J., Chhugani J., Kim C., Satish N., Lin M. C., Manocha D., Dubey P.: Clearpath: Highly parallel collision avoidance for multi-agent simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (August 2009), pp. 177–187.

[GDAB*17] Garcia-Dorado I., Aliaga D. G., Bhalachandran S., Schmid P., Niyogi D.: Fast weather simulation for inverse procedural design of 3d urban models. *ACM Transactions on Graphics (TOG) 36*, 2 (2017), 21:1–21:1.

[GDGAVU14] Garcia-Dorado I., Aliaga D. G., Ukkusuri S. V.: Designing large-scale interactive traffic animations for urban modeling. *Computer Graphics Forum 33*, 2 (May 2014), 411–420.

[GJBP96] Gardner K., Johnson T., Buchan K., PHARAOH T.: Developing a pedestrian strategy for london. In *Proceedings of Seminar B at the 24TH European Transport Forum of Transport Policy and Its Implementation*, Brunel University, England, 2–6 September, 1996.

[GRS95] Gilks W. R., Richardson S., Spiegelhalter D.: *Markov Chain Monte Carlo in Practice*. CRC Press, Boca Raton, FL, 1995.

[HAL63] Hall E. T.: A system for the notation of proxemic behavior. *American Anthropologist 65*, 5 (1963), 1003–1026.

[Hal66] Hall E. T.: *The Hidden Dimension*. Garden City, NY: Doubleday, *609* (1966).

[Has70] Hastings W. K.: Monte Carlo sampling methods using Markov chains and their applications. *Biometrika 57*, *1* (1970), 97–109.

[HFV00] Helbing D., Farkas I., Vicsek T.: Simulating dynamical features of escape panic. *Nature 407*, 6803 (September 2000), 487–490.

[HJ09] Helbing D., Johansson A.: *Pedestrian, Crowd and Evacuation Dynamics*. Springer New York, New York, NY (2009), pp. 6476–6495.

[HLB*17] Huang H., Lin N. C., Barrett L., Springer D., Wang H. C., Pomplun M., Yu L. F.: Automatic optimization of wayfinding design. *IEEE Transactions on Visualization and Computer Graphics 24*, 9 (2017), 2516–2530.

[HUB*15] Haworth B., Usman M., Berseth G., Kapadia M., Faloutsos P.: Evaluating and optimizing level of service for crowd evacuations. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games* (2015), ACM, pp. 91–96.

[HUB*16] Haworth B., Usman M., Berseth G., Khayatkhoei M., Kapadia M., Faloutsos P.: Towards computer assisted crowd aware architectural design. In *CHI EA '16: Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2016), ACM, pp. 2119–2125.

[HUB*17] Haworth B., Usman M., Berseth G., Khayatkhoei M., Kapadia M., Faloutsos P.: Code: Crowd-optimized design of environments. *Computer Animation and Virtual Worlds 28*, 6 (2017), e1749.

[Hug02] Hughes R. L.: A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological 36*, 6 (July 2002), 507–535.

[KGM13] Kim S., Guy S. J., Manocha D.: Velocity-based modeling of physical interactions in multi-agent simulations. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2013), ACM, pp. 125–133.

[KPAB15] Kapadia M., Pelechano N., Allbeck J., Badler N.: Virtual crowds: Steps toward behavioral realism. *Synthesis Lectures on Visual Computing 7*, 4 (2015), 1–270.

[LJK*12] Lemercier S., Jelic A., Kulpa R., Hua J., Fehrenbach J., Degond P., Appert-Rolland C., Donikian S., Pettré J.: Realistic following behaviors for crowd simulation. In *Computer Graphics Forum* (2012), Vol. 31, pp. 489–498.

[LPH*17] Liu W., Pavlovic V., Hu K., Faloutsos P., Yoon S., Kapadia M.: Characterizing the relationship between environment layout and crowd movement using machine learning. In *MIG '17: Proceedings of the 10th International Conference on Motion in Games* (New York, NY, USA, 2017), ACM, pp. 2:1–2:6.

[LSWW11] Lipp M., Scherzer D., Wonka P., Wimmer M.: Interactive modeling of city layouts using layers of procedural content. *Computer Graphics Forum (Proceedings EG 2011) 30*, 2 (April 2011), 345–354.

[MCG17] Moura F., Cambra P., Gonçalves A. B.: Measuring walkability for distinct pedestrian groups with a participatory assessment method: A case study in lisbon. *Landscape and Urban Planning 157* (2017), 282–296.

[MCTK16] Mac T. T., Copot C., Tran D. T., Keyser R. D.: Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems 86* (2016), 13–28.

[MRR*53] Metropolis N., Rosenbluth A. W., Rosenbluth M. N., Teller A. H., Teller E.: Equation of state calculations by fast

computing machines. *The Journal of Chemical Physics 21*, 6 (1953), 1087–1092.

[MZWVG07] Müller P., Zeng G., Wonka P., Van Gool L.: Image-based procedural modeling of facades. *ACM Transactions on Graphics (TOG) 26*, 3 (2007), 85.

[NGDA15] Nishida G., Garcia-Dorado I., Aliaga D.: Example-driven procedural urban roads. In *Computer Graphics Forum* (2015), Wiley Online Library.

[NGDA*16] Nishida G., Garcia-Dorado I., Aliaga D. G., Benes B., Bousseau A.: Interactive sketching of urban procedural models. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 130.

[NYF*15] Niu Y., Yang H., Fu J., Che X., Shui B., Zhang Y.: *Evacuation Simulation Incorporating Safety Signs and Information Sharing*. Springer International Publishing, Cham (2015), pp. 240–251.

[PAB07] Pelechano N., Allbeck J. M., Badler N. I.: Controlling individual agents in high-density crowd simulation. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 99–108.

[PAKB17] Pelechano N., Allbeck J. M., Kapadia M., Badler N. I.: *Simulating Heterogeneous Crowds with Interactive Behaviors*. Taylor and Francis Group, Boca Raton, FL, 2017.

[PF16] Pelechano N., Fuentes C.: Hierarchical path-finding for navigation meshes (HNA*). *Computer Graphics 59*, *C* (October 2016), 68–78.

[PGT08] Pettré J., Grillon H., Thalmann D.: Crowds of moving objects: Navigation planning and simulation. In *SIGGRAPH '08: ACM SIGGRAPH Classes* (2008), p. 54.

[PM01] Parish Y. I. H., Müller P.: Procedural modeling of cities. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), ACM, pp. 301–308.

[Rey87] Reynolds C. W.: Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics 21*, 4 (July 1987), 25–34.

[SBM*10] Stava O., Benes B., Mech R., Aliaga D. G., Kristof P.: Inverse procedural modeling by automatic generation of l-systems. *Computer Graphics Forum 29*, 2 (2010), 665–674.

[Sou05] Southworth M.: Designing the walkable city. *Journal of Urban Planning and Development 131*, 4 (2005), 246–257.

[SW86] Swendsen R. H., Wang J.-S.: Replica monte carlo simulation of spin-glasses. *Physical Review Letters 57*, 21 (1986), 2607.

[SWL11] Sewall J., Wilkie D., Lin M. C.: Interactive hybrid simulation of large-scale traffic. *ACM Transactions on Graphics 30*, 6 (December 2011), 135:1–135:12.

[TB16] Taal F. C., Bidarra R.: Procedural generation of traffic signs. In *UDMV '16: Proceedings of the Eurographics Workshop on Urban Data Modelling and Visualisation* (Goslar Germany, Germany, 2016), Eurographics Association, pp. 17–23.

[TCP06] Treuille A., Cooper S., Popović Z.: Continuum crowds. *ACM Transactions on Graphics 25*, 3 (July 2006), 1160–1168.

[TLL*11] Talton J. O., Lou Y., Lesser S., Duke J., Měch R., Koltun V.: Metropolis procedural modeling. *ACM Transactions on Graphics 30*, 2 (April 2011), 11:1–11:14.

[TM13] Thalmann D., Musse S. R.: *Crowd Simulation*. Springer-Verlag, London, 2013.

[Tor15] Torrens P. M.: Intertwining agents and environments. *Environmental Earth Sciences 74*, 10 (2015), 7117–7131.

[VAW*09] Vanegas C. A., Aliaga D. G., Wonka P., Müller P., Waddell P., Watson B.: Modeling the Appearance and Behavior of Urban Spaces. In *Eurographics 2009 - State of the Art Reports* (2009), M. Pauly and G. Greiner (Eds.), The Eurographics Association.

[vdBLM08] van den Berg J., Lin M., Manocha D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In *Proceedings of IEEE International Conference on Robotics and Automation* (May 2008), pp. 1928–1935.

[VGDA*12] Vanegas C. A., Garcia-Dorado I., Aliaga D. G., Benes B., Waddell P.: Inverse design of urban procedural models. *ACM Transactions on Graphics 31*, 6 (November 2012), 168:1–168:11.

[VKW*12] Vanegas C. A., Kelly T., Weber B., Halatsch J., Aliaga D. G., Müller P.: Procedural generation of parcels in urban modeling. *Computer Graphics Forum 31*, 2pt3 (May 2012), 681–690.

[vTTK*16] van Toll W., Triesscheijn R., Kallmann M., Oliva R., Pelechano N., Pettré J., Geraerts R.: A comparative study of navigation meshes. In *MIG '16: Proceedings of the 9th International Conference on Motion in Games* (New York, NY, USA, 2016), ACM, pp. 91–100.

[WWSR03] Wonka P., Wimmer M., Sillion F., Ribarsky W.: Instant architecture. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 669–677.

[YMC*05] Yersin B., Maim J., Ciechomski P., Schertenleib S., Thalmann D.: Steering a virtual crowd based on a semantically augmented navigation graph. In *Proceedings of 1st International Workshop on Crowd Simulation (V-CROWDS'05)* (November 2005), pp. 169–178.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Video S1