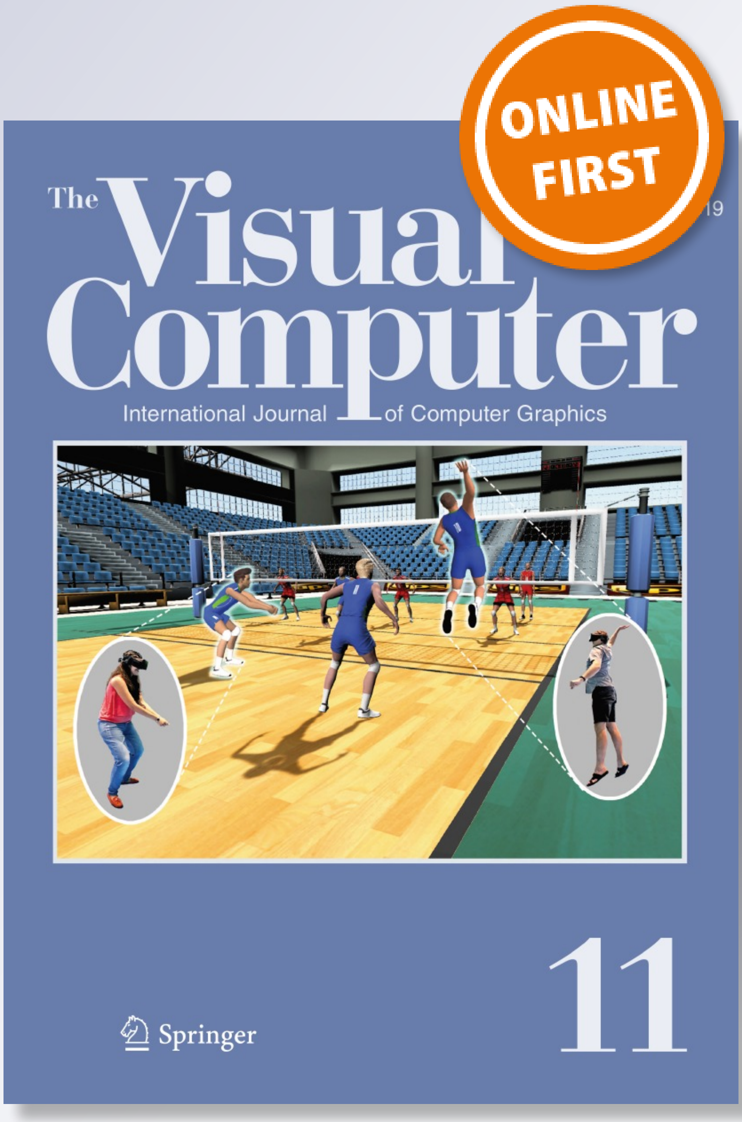
Generation of crowd arrival and destination locations/times in complex transit facilities

Brian Ricks, Andrew Dobson, Athanasios Krontiris, Kostas Bekris, Mubbasir Kapadia & Fred Roberts

The Visual Computer
International Journal of Computer Graphics

ISSN 0178-2789

Vis Comput
DOI 10.1007/s00371-019-01761-z



The Visual Computer
https://doi.org/10.1007/s00371-019-01761-z

ORIGINAL ARTICLE

Generation of crowd arrival and destination locations/times in complex transit facilities

Brian Ricks[1] · Andrew Dobson[2] · Athanasios Krontiris[3] · Kostas Bekris[4] · Mubbasir Kapadia[4] · Fred Roberts[4]

Abstract
In order to simulate virtual agents in the replica of a real facility across a long time span, a crowd simulation engine needs a list of agent arrival and destination locations and times that reflect those seen in the actual facility. Working together with a major metropolitan transportation authority, we propose a specification that can be used to procedurally generate this information. This specification is both uniquely compact and expressive—compact enough to mirror the mental model of building managers and expressive enough to handle the wide variety of crowds seen in real urban environments. We also propose a procedural algorithm for generating tens of thousands of high-level agent paths from this specification. This algorithm allows our specification to be used with traditional crowd simulation obstacle avoidance algorithms while still maintaining the realism required for the complex, real-world simulations of a transit facility. Our evaluation with industry professionals shows that our approach is intuitive and provides controls at the right level of detail to be used in large facilities (200,000+ people/day).

Keywords Crowd simulation · Crowd generation · Building simulation

1 Introduction

Crowd simulation has long been seen as a means of improving the quality of a building by increasing the flow of pedestrian traffic on a day-to-day basis or by reducing the chance of injury or death in an emergency. With these goals in mind, we engaged in a multi-year collaboration with one of the busiest transportation facilities in the world to model and simulate the pedestrian traffic within their structure. Based on our experience with previous crowd simulation research, we knew that we could rely on robust obstacle avoidance algorithms and global navigation algorithms (see our previous work section). In practice, these algorithms did indeed provide the agent motion planning needed. However, due to the size of the structure and the length of time to be simulated, we needed a new procedural generation method to create the arrival and destination locations and times for the large and diverse set of people that use such a large transit facility. Other researchers have studied the generation of high-level crowd paths to some degree (see, for example, Rogla et al.'s [1] referring to a similar problem to this as "Procedural Crowd Generation"). Unlike previous recent work in this area, our work is uniquely focused on generating crowd paths that match the arrival and departure rates of multiple modes of mass transportation in a complex indoor structure. Additionally, our specification for this procedural generation is the result of a multi-year collaboration and is designed to match the mental model of industry professionals. In this work, we discuss our approach to (1) specifying the arrival and destination places and times for large crowds, and (2) generating individual arrival and destination information based on this specification. Once we generated these paths, we then used traditional global path planning and obstacle avoidance algorithms to simulate crowds in a large structure for our industry partners.

Specific challenges in generating this specification and attendant algorithm included:

✉ Brian Ricks
bricks@unomaha.edu

1 University of Nebraska at Omaha, 6001 Dodge Street, Omaha, NE 68182-0500, USA

2 California Department of Healthcare Services, California, USA

3 Samsung Semiconductor, Inc., San Jose, California, USA

4 Rutgers, The State University of New Jersey, Piscataway, USA

- Creating a specification that mirrored the mental model of the building designers and managers.
- Capturing the wide range of ways people arrive and leave a facility, the wide range of demographics seen in a facility, and the wide range of vendors seen in a facility while keeping our specification compact.
- Allowing for global changes in behavior (such as evacuations and delays in departures).
- Handling the mass arrivals and departures commonly seen with mass transit.
- Making all the above specification components time-specific to capture the changes seen in crowds throughout the day.

The contributions of this work are (1) our specification and (2) our algorithm for generating arrival and destination information that resolves these challenges. We validate our work by documenting our feedback from our industry partners and the results of an informal user study.

2 Related work

In working with industry collaborators on this lengthy simulation project, we relied heavily on previous research both in the crowd simulation engine we used and as we developed our novel procedural generation specification and algorithm.

2.1 Crowd simulation research

After we generate the arrival and destination times and locations of our virtual agents, our algorithm relies on a traditional crowd simulation algorithm to simulate agents in a virtual replica of our structure. To do this, we relied on traditional obstacle avoidance techniques. Our simulation needed to run at the detail of virtual agents doing obstacle avoidance (as opposed to non-simulation techniques such as [2]) since the crowds in our facility created large lines. The presence of these queues would force people to take longer routes to their destinations, thus changing the time it took to get to where they were going. These dynamics were an inherent part of the facility's nature, and needed to modeled at the level of individual obstacle avoidance.

Most of obstacle avoidance techniques grew out of Reynolds' work on flocks and herds [3] and Helbing and Molnar's [4] social forces crowd model. Fiorini and Shiller's [5] velocity-based approach (notably refined by reciprocal velocity obstacles [6]) significantly advanced the field closer to where it is today.

Recent years continue to see improvements in crowd simulation algorithms. Wolinski et al. [7] created extremely precise local obstacle avoidance using probabilistic motion prediction, and Lu et al. [8] improved the performance of potential field-based crowds. Similarly, improvements have been seen with agents that realistically turn corners [9] and show appropriate etiquette when opening doors [10].

Other work focuses on agent-grouping behavior. Schuerman et al.'s work [11] added meta-data to locations in a structure or to a group of agents to avoid agent stalling and maintaining coherent agent groups. Other approaches include Kapadia et al.'s [12] framework focused on multi-agent scenarios.

At an even higher level, work has also been done at the level of agent desires or behaviors. Notable among these is Li and Allbeck's [13] work that gives agents social roles. Our underlying behavioral engine is based on the work of Krontiris et al. [14].

2.2 Large crowds

Leveraging early work in crowd simulation, researchers have used many of the above techniques to simulate crowds in large areas. Much of this research has focused on exterior environments. One of the clearest examples of this is that done in the area of crowd patches. Yersins et al. [15] put "crowd patches" together to populate a potentially infinite exterior environment made of local, pre-computed crowd simulations. This approach was further refined by Jordao et al. [16,17]. Another example of this kind of work is that done by de Paiva et al. [18].

A more recent exterior-focused crowd simulation work that more closely aligns with our contribution is that by Rogla et al. [1]. In this work, the authors combine city generation with crowd generation to both create a city and populate it. Their work successfully fills the gap often seen in video games between main characters that "tend to have rich behaviors manually defined using techniques such as Behavior Trees" and "background characters [that] tend to just use simple simulation methods." Their approach follows three major steps: generation of the city population, generation of each individual agenda, and then the actual simulation. Our approaches could be considered similar in that we also generate a population for our facility, give those agents goals, and then run our actual simulation. However, our approach to each of these steps is quite divergent due to the difference in the focus of work. Our contribution is focused on generating paths that match the complexity of people commuting between different modes of mass transit in multistory structures as opposed to the movement of people in an exterior environment. We particularly recommend Rogla et al.'s work to those interested in crowds for games, both for its academic contributions and its extensive review of crowd simulation in commercial games.

An example of work specifically designed for the interior of a real building is that done by Shau and Terzopoulos [19]. This work modeled the flow of pedestrians in a section of New

York's original Pennsylvania Station. Although this work puts crowds in a large structure, similar to our work, its focus is more on creating a virtual reality environment, not generating the arrival and destination times and locations for tens of thousands of agents.

Other work that has some similarity to our work includes work that gives semantic information to objects and locations. In addition to much of the work already cited, early work by Kallmann and Thalmann [20] takes a fine-detailed approach to semantic labeling. Additionally, work by Jorgensen [21] and Jorgensen and Lamarche [22] that takes into account scheduling activities of agents shares some themes to our work, although the scopes of our problem and approach are much different than theirs.

See [23,24] or [25] for a more extensive treatment of the vast field of crowd simulation.

2.2.1 Commercial crowd software

Commercial software, such as Massive [26], Golaem [27], Maya [28], and Houdini [29], are simulation-centric tools that animators use to author the responses of an autonomous agent to external stimuli and tweak simulation parameters to mold the emergent crowd behavior to conform to the required visual effect. These packages connect the environmental design and crowd simulation pieces into congruent pieces of software. Other software, such as Legion [30], have been written with building simulation in mind.

Even after entering into discussions with at least one commercial simulation vendor, our industrial partners asked us to create a custom simulation. The motivations for this included the need for the simulation to be tailored specifically to the behavior of the patrons in their building, the complexity of agents arriving and departing on different mass transit modalities, and the need for custom, tunable what-if scenarios.

3 Arrival and departure specification

Our crowd generation specification is based on our interactions with engineers and managers of one of the world's largest transit facilities located in the Eastern USA. The main pattern we observed in these practitioners was that they have a clear sense of the main groups of people in their buildings. These included the commuters that quickly came and went in addition to people who stayed in the building longer including employees, security, and the homeless. Similarly, these managers had a clear sense of how behaviors in a building changed over time. During the workweek they clearly delineated the pre-commute, morning commute, midday lull, evening crush, and then late evening behaviors. These experts could also readily explain how crowd behavior changed with certain holidays or events in the surrounding city. However, these descriptions were purely qualitative. Fortunately, it gave us a clear sense of the specification granularity that would work well with these managers—they thought about their building in terms of certain demographics and certain times.

As we visiting the facility, it became clear that determining the scope for crowd generation would be a non-trivial task. People arrived at the facility in multiple modalities, including on foot, in multiple modes of personal transportation, and in multiple modes of mass transportation. As a result, there were about two hundred places where people could arrive and depart across different floors. The pedestrian and personal transportation entrances and exits tended to have steady rates of arrival and departure. On the other hand, the mass transit entrances and exits often had long lines of people waiting to depart followed by a large rush of arrivals. In addition, some of the mass transit locations were for long distance travel; the patrons waiting at these departure points frequently arrived long before their scheduled departure times and brought multiple pieces of luggage.

Based on these interactions, we propose a compact yet expressive enhanced crowd specification. The specification includes the following:

- Location points of agents with arrivals that have either a steady rate or scheduled, large arrivals.
- Location points of agents with departures that have either continuous egress or queues followed by large departures.
- Distinct demographics within the structure with each arrival location spawning specific distributions of each demographic.
- Non-departure and destination locations that affect agents' behavior within the structure.
- Customizable global changes in behavior such as fire alarms.
- Every aspect of the specification being able to change with time.

```
-
  Food
   AttractionStrength:
      {0, .5},
      {660,3},
      {780, 1},
      {960,4}
      {1080,1.5}
 -
  Restrooms
   AttractionStrength:
      {0, 1},
```

Fig. 1 Pseudocode of attractor information in the global sub-specification for a simple building. This lists two attractor types: food vendors and restrooms. The food vendors have a low attraction at the start of the simulation (min 0) that goes up around lunch time, drops down in the afternoon, and then goes up again around dinner time. The restrooms in this example have a constant attraction strength

To include all of these parts, our specification is designed in three main sections. First, the global sub-specification defines the global variables for the entire facility. Easy modification for the entire structure allows for simple “what-if” scenario testing and modeling events with specific global behaviors (vacation days, busy shopping days, etc.). Second, the local sub-specification allows individual areas of the structure to be defined as having the properties of a global variable with additional, more specific information. This allows the user to give a terminal exact information about arrival times or a food vendor details about relative popularity. Third, the high-level changes sub-specification allows time-based events to be added to the simulation, for example, a global delay in the departure of trains starting at a certain time or weather-related slippery conditions that slow pedestrian movements in the afternoon. We discuss each of these sub-specifications in turn.

3.1 Global sub-specification

The first part of our specification is the global sub-specification. This provides general information about the attractors, people types, and origin types in the structure. Later, in the local sub-specification, individual parts of the structure will define their specific information in terms of this global sub-specification plus additional details. Each part of the global sub-specification was chosen to match the mental model of industry professionals, while still providing information needed for a crowd simulation algorithm.

The first kind of global information is a list of **attractor types** seen in the facility (see Fig. 1). An attractor is a location in the building that can affect the behavior of people in the building on their way from their arrival location to their destination location. Attractors are integrated into the behavioral aspects of the crowd simulation algorithm at run time. Attraction types might include retail locations, food vending, ticket booths, restrooms, offices, and meeting areas. Each of the global types can be annotated to indicate how interest in these locations changes throughout the day. For example, food locations are more influential around meal time. As noted earlier, the ability to vary the attraction of inter-facility locations was a critical expectation of our expert collaborators. Later, individual locations in our specification can be labeled as being one of these attraction types, along with other local details.

The second kind of global information is a list of major **people types** that move through the facility (see Fig. 2). Such demographic types might include commuters, shoppers, employees, and security personnel. Each major people type includes information about how long they are expected to stay in the facility and how interested they are in each global attraction type. Thus, commuters may stay in the building for shorter periods of time and are highly interested in buying tickets, while employees will stay much longer and spend most of their time in their offices. Additional demographic information can be added about each major agent type that uses the facility, including a distribution over the expected walking speed and the percent that have luggage or a disability that will affect mobility. Lastly, each major agent type is labeled as representing agents that either (A) return from where they entered (stay agents), or as (B) representing agents that leave at a different place than where they entered (through agents). In a transit facility, commuters will generally enter and leave at different locations (through agents) while employees will generally come and go from the same location (stay agents). This differentiation will be discussed at length in our algorithm section.

The third kind of global information is a list of major **origin/destination types** (see Fig. 3). These can include designations such as bus gates, airline terminals, subway entrances/exits, parking, and sidewalk entrances/exits. Each major origin/destination type also can include annotations about what percent of the people who enter there are from each major agent type. For through agent types (as described above), there is also a distribution over the destinations those agents will choose.

3.2 Local sub-specification

The local sub-specification comprises individual details about specific locations within the facility. By giving details in terms of global information plus local variations from that global information, it is easy to create what-if scenarios that change global information about a facility while still preserving the unique characteristics of each part of a building. The

```
-
 Commuters
  PercentThroughAgents: .95
  AttractorStrengths:
   Food: 0.0, .5
   Restrooms: 0.0, 1.0
  Walking Speed: .9, 1.1
  PercentWithDisability: .01
  PercentWithLugage: .02
-
 Employees
  PercentThroughAgents: .1
  AttractionStrengths:
   Food: 0.0, .1
   Restrooms: 0.1, .5
  Walking Speed: .8, .9
  PercentWithDisability: .01
  PercentWithLugage: .01
```

Fig. 2 Pseudocode of people type information in the global sub-specification for a simple building. This lists two people types: commuters and employees. The commuters are almost all through agents (they arrive and leave at different locations) while employees are not (meaning they are mostly stay agents, so they arrive and leave at the same locations). Both people types have their own distributions for their interest in attractor types and walking speed (specified as a minimum and maximum value) as well as percent disabled and percent with luggage, which affect walking speed, etc

```
-
  Name: Sidewalks
  AgentTypes:
   Employees: 0.01
   Commuters: 0.99
  ThroughAgentDestinationRatios:
      Bus: 0.4
      Subway: 0.45
      Sidewalk: 0.05
 -
  Name: Subways
  AgentTypes:
   Employees: 0.02
   Commuters: 0.98
  ThroughAgentDestinationRatios:
      Bus: 0.3
      Subway: 0.1
      Sidewalk: 0.6
 -
  Name: Buses
  AgentTypes:
   Employees: 0.03
   Commuters: 0.97
  ThroughAgentDestinationRatios:
      Bus: 0.05
      Subway: 0.5
      Sidewalk: 0.45
```

Fig. 3 Pseudocode of origin type information in the global sub-specification for a simple building. This lists three origin types: buses, subways, and sidewalks. Each origin type specifies a distribution of people types that arrive there and, of those that are through agents, their destination ratios

local sub-specification includes details for individual attractors and individual origin/destinations (see Fig. 4).

First, **local attractor information** specifies the type of attraction a certain area has. For example, an area may be assigned the type of retail outlet, a restaurant, etc., if such a type exists in the global sub-specification. In addition to choosing a global type, our specification allows for location-specific distributions over how long people will stay in a location and how many people can fit into a location before a queue forms outside the location.

```
#LocalInformation
-
  Name: BusTerminal10
 Origins:
   Type: Buses
   TimesAsSpans=False
   Arrivals:
    {100,65.0}
   Departures:
    {200,70.0}
-
  Name: Restaurant1
  Attractions:
   AttractionType:Food
-
  Name: MainRestrooms
  Attractions:
   AttractionType:Restroom
-
  Name: Sidewalk16
 Origins:
   Type: Sidewalks
   TimesAsSpans=True
   Arrivals:
    {0,50.0}
    {60,60.0}
    {120,65.0}
   Departures:
      DeparturePriority: 1
```

Fig. 4 Example of the local part of our specification given as pseudocode. Notice the list of individual origins and attractors

```
#GlobalChanges
-
  Type: DepartureDelay
   AffectedTypes:
    Type: Buses
   Start: 1020
   End: 1035
```

Fig. 5 Example of a high-level change given as pseudocode. This change gives a delay in all bus departures starting at 5 pm (17:00) and lasting 15 min

Second, for each specific area that acts as an origin/destination point, the user can indicate which global **origin/destination type** this location is. In addition to this global information, details are provided about how people are expected to arrive and leave at the location. We do this for two main arrival rate types: steady flow and bulk arrivals. For exterior entrances that have steady rates of arrival, such as parking lots and sidewalk entrances, information is provided in terms of people/minute, which can change throughout the day. For example, 10 people arrive per minute from the parking lot from midnight to 6 am, 30 people arrive per minute from 6 am to 8 am, etc. For mass transit locations that have bulk arrivals, arrivals are specified in terms of people arriving en masse at certain times. For example, a bus arrives at 7:12 with 50 people, a plane arrives at 8:33 with 128 people, and so on.

Also included in the local information are details about how people leave at a location. Similar to the local arrival specification, there are two types of departure rates: steady flow and bulk departures. For departure locations with steady flow, the specification does not indicate how many people leave per minute, since that is calculated algorithmically. Instead, the user can specify a destination priority, or how likely an agent is to choose this specific destination when choosing among destinations of the same type. For example, our observation is that certain street exits are far more popular than others. The destination priority captures these preferences. If agents depart from a location in bulk, then departures are specified by time and with the maximum number of people that could leave. For example, a bus departs at 11:30 with up to 75 people, a plane leaves at 6:00 with up to 200 people, etc.

3.3 High-level changes sub-specification

The above specifications give us the information needed to run a crowd simulation. However, all of this information is too detailed for a facility engineer to change to test a what-if scenario such as ‘what if all departures are delayed by 15 min starting at noon?.’ Additionally, it does not match the higher-level mental model we observed in industry experts. Since one of the main use cases of this enhanced building specification is performing what-if testing, we added the capacity to indicate changes at a higher-level that match the needs of these professionals (see Fig. 5).

For example, our experience is that managers and engineers are interested in how movements within the building will change if all mass transit departures are delayed by 15 min. This can be specified by altering each mass transit arrival point and adding a delay; however, this is tedious and not at the level that the managers and engineers are thinking. Thus, our specification also allows a user to include specific information about global changes to crowd behavior. These changes (detailed later) include:

- Global changes in departures by transit type.
- Global changes in condition (such as rain and snow) that affect people's walking speed.
- Emergency conditions, such as an evacuation.

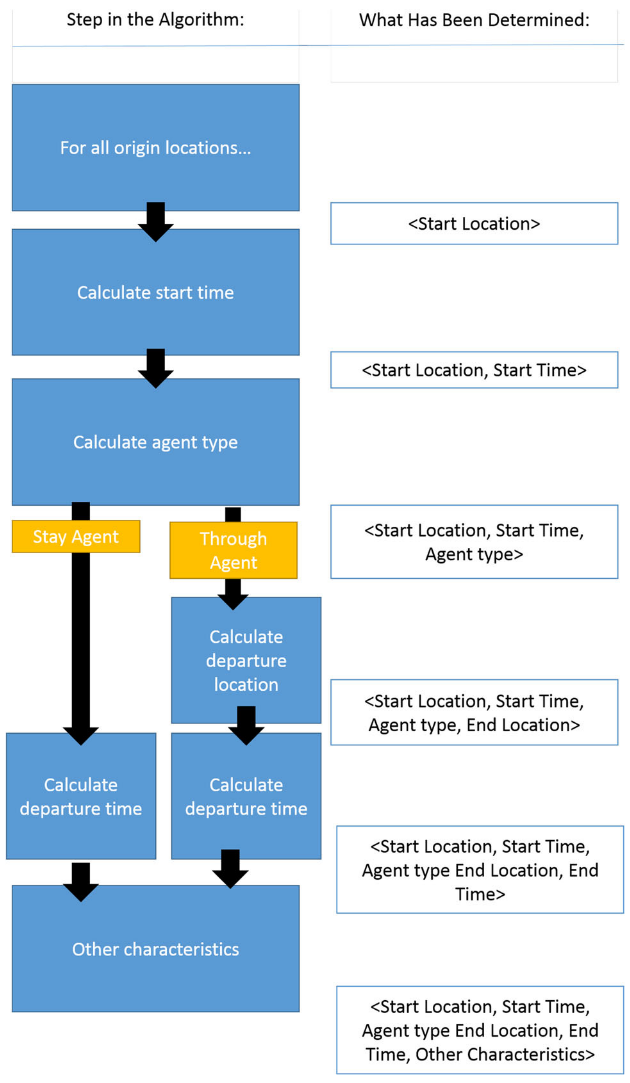

Step in the Algorithm:
What Has Been Determined:
For all origin locations...
<Start Location>
Calculate start time
<Start Location, Start Time>
Calculate agent type
Stay Agent
Through Agent
<Start Location, Start Time, Agent type>
Calculate departure location
<Start Location, Start Time, Agent type, End Location>
Calculate departure time
Calculate departure time
<Start Location, Start Time, Agent type End Location, End Time>
Other characteristics
<Start Location, Start Time, Agent type End Location, End Time, Other Characteristics>

Fig. 6 This figure shows the process of taking our enhanced crowd specification and generating a list of agent paths for a crowd simulation to model. (Left) The algorithmic steps that happen in sequence. (Right) The information that is known once each step is completed

4 Calculating agent paths

Our first major contribution is our crowd and building specification as described above. This specification is not useful without an algorithm to generate the arrival and destination locations and times for agents. We now give the details of our second contribution, the algorithm that generates these high-level paths. By generating these paths, traditional crowd simulation algorithms can be used to simulate the movement of people within large, complex structures.

The steps taken by this algorithm are detailed in Fig. 6. The goal of these steps is to create high-level agent paths that contain the following information: arrival time, arrival location, agent type, desired departure time, departure location, and agent information. In other words, our algorithm implements a function f such that f takes a crowd and building specification $sp \in SP$ as its input and produces a set of paths $P' \in P$, where $P = \{\text{arrival}_t \times \text{arrival}_l \times \text{agent}_{\text{type}} \times \text{departure}_t \times \text{departure}_l\}$. The resulting set of paths P' can then be used as the input into a traditional crowd simulation algorithm.

4.1 Determining arrival times

The first step is to determine the arrival times of agents. To do this, our algorithms loops over each room to see if it has origin information. If it does, then the arrival information is used to create arrival times.

How arrival times are chosen depends upon the arrival type. If the origin has a constant arrival rate, then our algorithm calculates the number of arrivals per time period. For example, the specification may indicate that 100 people arrive per hour. In this case, the algorithm randomly chooses 100 times within that hour for people to arrive. This effectively does a uniform sampling of arrivals, stratified by the hour. On the other hand, if this origin has bulk arrivals (for example, from a bus, train, or plane), then the specified number of arrivals is created at the given arrival time. Arrivals are spaced slightly (by a second or two) to prevent agents being created on top of each other.

At this point in the process, we have a list of paths with the arrival times and arrival locations, i.e., $P' \subset \{\text{arrival}_t \times \text{arrival}_l\}$.

4.2 Determining agent types and departure times

Once we have a list of high-level paths with the arrival time and arrival location, the algorithm calculates agent types. Each arrival place has local origin information (as described above) that specifies what global type best describes that local origin. For example, an origin can be a bus gate, a subway station, a sidewalk entrance, etc. Since each global type includes information about the distribution of agent types that arrive at that global type, the algorithm samples that distribution to assign the agent type. This is done using a simple weighted sampling.

At this point, our algorithm has a list of paths with the arrival time, arrival place, and agent type of each agent, i.e., $P' \subset \{\text{arrival}_t \times \text{arrival}_l \times \text{agent}_{\text{type}}\}$. The algorithm now branches depending on whether or not the agent type is a stay agent or a through agent (as defined previously).

4.2.1 Stay agents

If the agent type chosen has the stay agent property (i.e., each agent of this agent type leaves where it enters, as is often the case of employees), then the algorithm can immediately assign the agent's destination location. In this case, it is the arrival location. The specification has a distribution of desired durations for each agent type if they are a stay agent. Thus, the algorithm samples the stay agent duration distribution to choose a desired duration. This assigned duration prevents stay agents from immediately turning around and leaving the facility and instead visiting the various attractors inside the facility.

At this point, for stay agents, the algorithm has a list of paths with arrival time, arrival place, agent type, desired duration, and destination location.

4.2.2 Through agents

Through agents require more work than stay agents since the algorithm has to determine where through agents will leave. This is a non-trivial process since many patrons will choose mass transit options that leave on a set schedule. This leads to two possible approaches. In our facility, rates of departure on mass transit were some of the best documented figures available for our model. This led to one possible solution, which was to work backward and start by calculating agent departure times and then calculating agent arrival times based on that figure. This may be the correct approach in a facility where agents primarily arrive at a constant rate, but this did not work in our case, as the majority of the patrons departing via mass transit also arrived via mass transit (e.g., arriving on one train and departing on another). Resolving this problem was non-trivial, but the resulting solution creates a very plausible and useful crowd simulation.

The first step in our ultimate solution was to sample the distribution of destination types for the current origin type and to treat the result as a *desired* departure time as opposed to a fixed departure time. If the destination type was where people leave at regular rates (such as a sidewalk entrance or parking garage), the exact destination location was chosen based on the destination priority information. If the agent chose to leave at a destination type that had bulk departures, the agent chose a specific destination by weighing each possible departure time. The weight was determined by comparing the bulk departure time to the agent's desired departure time. If the bulk departure time was close to the agent's desired departure time, then the weight was high; if the departure time was not close to the agent's desired departure time, then the weight was low. Once a bulk departure was chosen, the agent adjusted its departure time to match the bulk departure time and set its desired destination to that location.

At this point in the algorithm, regardless of whether the agent is a stay agent or a through agent, each agent now had an agent type, arrival time, arrival location, departure location, and desired departure time, e.g., $P' \subset \{\text{arrival}_t \times \text{arrival}_l \times \text{agent}_{\text{type}} \times \text{departure}_t \times \text{departure}_l\}$.

4.3 Other variables

Additional information for the agent can be calculated using simple distributions. For example, each agent type has a distribution over the probability of an agent being disabled (i.e., walking slower and requiring the use of elevators), having luggage, and general walking speed. Similarly, each agent type has a distribution for interest for each attraction type. Each of these distributions is sampled to give each individual agent specific desires to visit attractions in the facility. It is then up to the behavioral algorithm used by the crowd simulation engine to use these at runtime.

4.4 High-level changes

As noted earlier, building managers and designers often want to do what-if analysis at a high level. For example, they may want to study what happens when all trains are delayed by an hour. This could be accomplished by manually altering all the destination information in the facility, but this would require tedious work. In a desire to simplify and match the mental model of the experts with whom we worked, we also provide the ability to make high-level or event-level changes to the specification. These high-level changes are also be taken into account during the agent creation process.

The first global change reflects a delay in all departures of a certain type. For example, an airport may have adverse weather that delays all flights in a given time period by an hour. This could be done in two different ways. It could be done by changing the departure times at each location and then running our path calculation algorithm. On the other hand, it could be done by calculating all the agent paths and then delaying all departures that fell within the specified delayed period. We found that this latter approach worked better since it better reflects what happens in a real delay situation.

Other global changes reflect changes in walking speeds (as seen on rainy/snowy days) or when an evacuation starts. Global changes in walking speed adjust the walking speed parameter on each agent. An evacuation tells the crowd simulation engine to change agents' destinations at a given time.

Combined, the steps translate our high-level specification into low-level information for a crowd engine.

4.5 Limitations

This approach is not a general purpose crowd simulation solution since it is designed for transit facilities. Other crowd simulation domains (consider highly social ones as in [32]) would probably find the arrival and destination contribution of this work most applicable to their work. Also, our model requires a large number of parameters—in our specific facility the hundreds of arrival and departure parameters were provided by the facility. In cases such information is not available, or the building is not a real one, such information will need to be generated. One such approach can be to follow the work of Rogla et al.'s [1] where building generation and agent generation are created simultaneously. This approach can be used to determine many of the parameters in the simulation (see Figs. 1, 2, 3, 4).

5 Validation, results, and informal user study

Due to our non-disclosure agreement, we cannot provide numeric details about the actual crowds inside the facility. Thus, in validating our results, we provide the expert feedback and an informal user study.

After the simulation ran, we compiled agent density results and created heat maps showing where the simulation predicted crowds were the most dense in the current facility layout (see Fig. 7). This allowed us to validate our model in three ways. First, the building stakeholders reviewed this data and gave us feedback about how the density data compared to their mental model of density in the real building. Second, we had footage from security cameras within the building and knew where the choke points were within in the facility. Third, we had spent extensive time within the facility ourselves and had begun to build our own mental model of what the crowd flow looked like in the real facility. All three of these validated the numeric results of our simulation.

At the end of this multi-year collaboration our partners were pleased with the results of our work. Specifically, they were impressed that we could create a specification that captured the dynamics of such a complex structure, that the specification was designed to match their mental model, and that it could model the many transportation modalities in their facility. They were also impressed with the crowd simulations we could produce from the results of our specification and algorithm and the analysis of their structure it allowed us to do together.

We further validated our specification with informal user studies. In the first study, we created an app for creating our specification. This was used by three graduate students, two of whom were familiar with crowd simulation and one of whom was familiar with building design. The graduate students were given tasks in designing a building and/or specifying crowd behavior. In the second study, this system was used by two undergraduate students, both of whom were unfamiliar with building design and crowd simulation. Both of these undergraduate students were given detailed tasks in designing and authoring crowds in a large facility.

Several results came out of these user studies that informed our work. Foremost, we learned that the specification should be broken down into clear components. This led to the clear delineation between the three sub-specifications described above (global, local, and high-level changes). These changes were also greeted warmly by our industry collaborators, as it was a closer mapping to their mental model. These informal user studies also showed that multiple people with different backgrounds could use our specification.

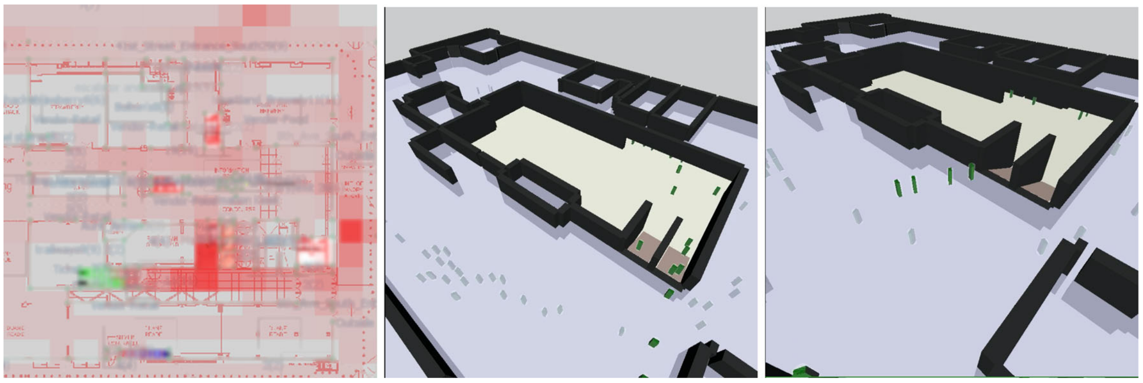


Fig. 7 Left: Heatmap of crowd densities within our facility. Similar heat maps were used to validate our simulation. Center and Right: Simple renderings of crowds specified using our crowd specification. These two examples show crowds at a staircase at two different times

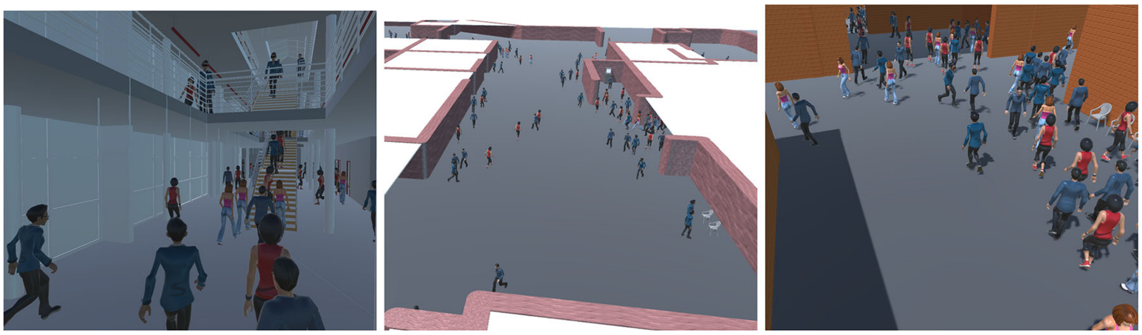


Fig. 8 Final renderings of crowds using the Unity [31] engine that show the complexity of the facility simulated

In terms of implementation, we used an RVO-based engine called PracSys [33] and is representative of other available crowd simulation engines. The behavioral engine we used was similar to that explained by Krontiris et al. [14]. Using our specification, we were able to load crowds into this vast facility (see Fig. 7). We then rendered our results together with geometry available from the commercial design tools (see Fig. 8) using the Unity [31] engine.

6 Future work

We have presented a specification and algorithm for procedurally generating crowd paths for a real transit facility. Looking forward, we see research opportunities in the area of environment optimization. As crowd simulations can provide quantitative results about the usability of a space, algorithms could be designed to optimize the arrivals and destinations locations, similar to how Feng et al. [34] adjust retail spaces. For example, our software could explore a space of different options and use results from a crowd simulation engine to update the building to the most efficient model found. Another possibility is that the software could suggest this improved layout to the users, who could use their expert knowledge to decide if the proposed decision will lead to the desired outcome.

Funding This study was funded in part by NSF #1718139 and by DHS #2009-ST-061-CCI002-06 to Rutgers University. All authors were in a contractual agreement with the transit facility mentioned that supported the researchers directly or funded students.

Compliance with ethical standards

Conflict of interest The authors declared that they have no other conflict of interest.

References

1. Rogla Pujalt, O., Pelechano Gómes, N., Patow, G.: Procedural crowd generation for semantically augmented virtual cities. In: CoRR (2018). arXiv:1811.10036
2. Testa, E., Barros, R.C., Musse, S.R.: Crowdest: a method for estimating (and not simulating) crowd evacuation parameters in generic environments. Vis. Comput. **35**(6), 1119–1130 (2019)
3. Reynolds, C.W.: Flocks, herds and schools: a distributed behavioral model. In: ACM Siggraph Computer Graphics, vol. 21, pp. 25–34. ACM (1987)
4. Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51**(5), 4282 (1995)
5. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. Int. J. Robot. Res. **17**(7), 760–772 (1998)
6. Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: IEEE International Conference on Robotics and Automation 2008, ICRA 2008, pp. 1928–1935. IEEE (2008)
7. Wolinski, D., Lin, M.C., Pettré, J.: WarpDriver: context-aware probabilistic motion prediction for crowd simulation. ACM Trans. Graph. **35**(6), 164:1–164:11 (2016)
8. Guanghui, L., Chen, L., Luo, W.: Real-time crowd simulation integrating potential fields and agent method. ACM Trans. Model. Comput. Simul. **26**(4), 1–16 (2016)
9. He, G., Jin, Y., Chen, Q., Liu, Z., Yue, W., Lu, X.-J.: Shadow obstacle model for realistic corner-turning behavior in crowd simulation. Front. Inf. Technol. Electron. Eng. **17**(3), 200–211 (2016)
10. Huang, W., Terzopoulos, D.: Door and doorway etiquette for virtual humans. IEEE Trans. Vis. Comput. Graph. (2018) https://doi.org/10.1109/TVCG.2018.2874050
11. Schuerman, M., Singh, S., Kapadia, M., Faloutsos, P.: Situation agents: agent-based externalized steering logic. Comput. Anim. Virtual Worlds **21**(3–4), 267–276 (2010)
12. Kapadia, M., Singh, S., Reinman, G., Faloutsos, P.: A behavior-authoring framework for multiactor simulations. IEEE Comput. Graph. Appl. **31**(6), 45–55 (2011)
13. Li, W.P., Allbeck, J.M: Populations with purpose. In: Motion in Games (2011)
14. Krontiris, A., Bekris, K.E., Kapadia, M.: Acumen: activity-centric crowd authoring using influence maps. In: Proceedings of the 29th International Conference on Computer Animation and Social Agents, CASA '16, pp. 61–69. ACM, New York (2016)

15. Yersin, B., Maïm, J., Pettré, J., Thalmann, D.: Crowd patches: populating large-scale virtual environments for real-time applications. In: SI3D (2009)
16. Jordao, K., Charalambous, P., Christie, M., Pettré, J., Cani, M.-P.: Crowd art: density and flow based crowd motion design. In: Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games, MIG '15, pp. 167–176. ACM, New York (2015)
17. Jordao, K., Pettré, J., Christie, M., Cani, M-P.: Crowd sculpting: a space-time sculpting method for populating virtual environments. Comput. Graph. Forum. (2014). https://doi.org/10.1111/cgf.12316
18. de Paiva, D.C., Vieira, R., Musse, S.R.: Ontology-based crowd simulation for normal life situations. In: International 2005 Computer Graphics, pp. 221–226 (2005)
19. Shao, W., Terzopoulos, D.: Autonomous pedestrians. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 19–28. ACM (2005)
20. Kallmann, M., Thalmann, D.: Modeling objects for interaction tasks. In: Arnaldi, B., Hégron, G. (eds.) Computer Animation and Simulation '98, pp. 73–86. Springer, Vienna (1999)
21. Jørgensen, C.-J.: Scheduling activities under spatial and temporal constraints to populate virtual urban environments. Theses, Université Rennes 1 (2015)
22. Jorgensen, C.-J., Lamarche, F.: Space and time constrained task scheduling for crowd simulation. Research report PI 2013 (2014)
23. Kapadia, M., Pelechano, N., Allbeck, J., Badler, N.: Virtual crowds: steps toward behavioral realism. Synth. Lect. Vis. Comput. Comput. Graph. Anim. Comput. Photogr. Imaging **7**(4), 1–270 (2015)
24. Pelechano, N., Allbeck, J.M., Kapadia, M., Balder, N.L.: Crowds with Interactive Behaviors. Taylor and Francis Group, London (2017)
25. Thalmann, D.: Crowd Simulation. Wiley, New York (2007)
26. Massive Software: Massive. http://massivesoftware.com (2019). Accessed 10 Oct 2019
27. Golaem: Golaem. http://golaem.com (2019). Accessed 10 Oct 2019
28. Autodesk: Maya. https://www.autodesk.com/products/maya/overview (2019). Accessed 10 Oct 2019
29. SideFX: Houdini. https://www.sidefx.com/ (2019). Accessed 10 Oct 2019
30. Bentley: Legion Software. https://www.bentley.com/en/products/brands/legion (2019). Accessed 10 Oct 2019
31. Unity3D: Unity. https://unity.com/ (2019). Accessed 10 Oct 2019
32. Ricks, B.C., Egbert, P.K.: A whole surface approach to crowd simulation on arbitrary topologies. IEEE Trans. Vis. Comput. Graph. **20**(2), 159–171 (2014)
33. Kimmel, A., Dobson, A., Littlefield, Z., Krontiris, A., Marble, J., Bekris, K.E.: Pracsys: an extensible architecture for composing motion controllers and planners. In: Simulation, Modeling, and Programming for Autonomous Robots, pp. 137–148. Springer (2012)
34. Feng, T., Yu, L.-F., Yeung, S.-K., Yin, K.K., Zhou, K.: Crowd-driven mid-scale layout design. ACM Trans. Graph. **35**(4), 132:1–132:14 (2016)

Brian Ricks is the director of the Bricks Lab at the University of Nebraska at Omaha, which focuses on crowd simulation and virtual worlds. He distinguishes himself by working directly with building managers and designers to identify fundamental research questions in the field of crowd simulation. His work has been published in Transactions on Visualization and Computer Graphics and the Visual Computer as well as in numerous conferences. His work is funded by multiple NSF awards.

Andrew Dobson received both his Bachelor's degree and Masters Degree in Computer Science from the University of Nevada, Reno. He received his Ph.D. at Rutgers University and did post-doctoral work at the University of Michigan.

Athanasios Krontiris received a Bachelor's degree in Computer Science from the University of Crete (2007). On August 2009 he joined the Department of Computer Science and Engineering at the University of Nevada, Reno (UNR) as a graduate student. In May 2011 he completed a Master's degree in Computer Science. On July 2012, he moved to Rutgers, the State University of New Jersey where he would complete his Ph.D. work under the supervision of Dr. Kostas Bekris.

Kostas Bekris received a Bachelor's degree in Computer Science from the University of Crete in 2001. He completed both his Master's (2004) and Doctoral (2008) degrees in Computer Science under the supervision of Prof. Lydia Kavraki. On July 2008 he joined the Department of Computer Science and Engineering at the University of Nevada, Reno (UNR) as an Assistant Professor. On July 2012, he moved to Rutgers University and joined the Computer Science department as an

Assistant Professor. Since July 2016, he is serving as Associate Professor in the same department.

Mubbasir Kapadia is the Director of the Intelligent Visual Interfaces Lab and an Assistant Professor in the Computer Science Department at Rutgers University. Previously, he was an Associate Research Scientist at Disney Research Zurich. Kapadia's research lies at the intersection of artificial intelligence, visual computing, and human–computer interaction, with a mission to develop intelligent visual interfaces to empower content creation for human-aware architectural design, digital storytelling, and serious games. Kapadia's research is funded by DARPA and NSF, and through generous support from industrial partners including Disney Research, and Unity Labs. He received his Ph.D. in Computer Science at UCLA.

Fred Roberts is a Distinguished Professor of Mathematics at Rutgers University, where he is a member of seven graduate faculties, in Computer Science, Mathematics, Operations Research, Computational Molecular Biology, BioMaPS (Interdisciplinary Ph.D. Program at the Interface between the Biological, Mathematical, and Physical Sciences), Industrial and Systems Engineering, and Education. He has served as Director of CCICADA (a U.S. Department of Homeland Security Center of Excellence) since its founding in 2009, and previously served as Director of the Center for Dynamic Data Analysis (DyDAn), the predecessor DHS University Center of Excellence to CCICADA, from 2006 to 2009.