Building Agent-Based Walking Models by Machine-Learning on Diverse Databases of Space-Time Trajectory Samples

Dr. Paul M. Torrens, Associate Professor and Director, Geosimulation Research Laboratory, Arizona State University, Tempe, AZ 85287-5302. Email: torrens at geosimulation dot com; Web: http://www.geosimulation.org

Mr. Xun Li, Geosimulation Research Laboratory, Arizona State University, Tempe, AZ 85287-5302. Email: xun.li@gmail.com

Dr. William Griffin, Center for Social Dynamics and Complexity, Arizona State University, Tempe, AZ 85287. Email: WILLIAM.GRIFFIN@asu.edu

Citation: Torrens, P.M., Li, X., Griffin, W.A. (2011). "Building agent-based walking models by machine-learning on diverse databases of space-time trajectory samples". *Transactions in Geographic Information Science*, 15(s1): 67-94; DOI: 10.1111/j.1467-9671.2011.01261.x

Abstract

We introduce a novel scheme for automatically deriving synthetic walking (locomotion) and movement (steering and avoidance) behavior in simulation from simple trajectory samples. We use a combination of observed and recorded real-world movement trajectory samples in conjunction with synthetic, agent-generated, movement as inputs to a machine-learning scheme. This scheme produces movement behavior for non-sampled scenarios in simulation, for applications that can differ widely from the original collection settings. It does this by benchmarking a simulated pedestrian's relative behavioral geography, local physical environment, and neighboring agent-pedestrians; using spatial analysis, spatial data access, classification, and clustering. The scheme then weights, trains, and tunes likely synthetic movement behavior, per-agent, per-location, per-time-step, and per-scenario. To prove its usefulness, we demonstrate the task of generating synthetic, non-sampled, agent-based pedestrian movement in simulated urban environments, where the scheme proves to be a useful substitute to traditional transition-driven methods for determining agent behavior. The potential broader applications of the scheme are numerous and include the design and delivery of location-based services, evaluation of architectures for mobile communications technologies, what-if experimentation in agent-based models with hypotheses that are informed or translated from data, and the construction of algorithms for extracting and annotating space-time paths in massive data-sets.

Keywords: machine learning; agent-based model; trajectory sample; data mining; geosimulation.

"*'Sometimes you repeat yourself, man.' 'It's in my nature'*." (Gibson 1984, p. 80)

1 Introduction

Walking and related locomotion is of significant interest for many domains, but many aspects of walking are difficult to experiment with in the real-world and can only be reliably explored through experimentation with computer models. To be useful, models need to be realistic. High-fidelity representation of human walking behavior is commonly desired and model-builders often use agent-based models (ABMs) as a mechanism for representing individual agency in simulation, but in many cases they specify the models with coarse, abstract representations of movement (Hughes 2003). One of the reasons for this is the difficulty of collecting individual behavioral data in natural contexts, particularly for situations in which many walkers are considered. *Behavioral* inference, in particular, requires time-consuming observation and coding (Schwartz and Schwartz 1955). Moreover, there are often many emergency scenarios for which it is unrealistic to collect data on the ground. Recent developments in mobile positioning systems have made this easier, enabling automatic reporting of people's positions from carried location-aware devices (Torrens 2010b), but significant challenges remain in reliably associating movement paths with the reasons *why* people move the way that they do. In essence, few pipelines exist to map walking *data* to *behavior*. Developers of agent-based models consequently find it difficult to derive informed rules for agent walking behavior *at the scale of individual agents* and this leaves them little alternative but to employ abstract, proxy representations of movement (Torrens 2010a).

In this paper, we introduce a scheme to resolve this problem by infusing movement simulations with automatically-derived movement behaviors derived from machine-learning on trajectory samples. We will demonstrate that individual walking behavior in models can be machine-learned from simple movement samples (snapshots of positioning and timing of locomotion). This provides an alternative to coarse models of movement. It also extends traditional agent-based transition schemes, which hardcode agents' movement rules into a model *a priori*. By contrast, our agents can develop their movement behavior dynamically, allowing greater model flexibility, because they learn movement anew, individually, for every agent, at every step, for every choice that they encounter in simulation. We also introduce a high-level behavior controller that allows model-users to specify agent-based transitions as Finite State Machines, adding further flexibility in repurposing trajectory data to new scenarios. To demonstrate the value of our approach, we will illustrate several application scenarios using different experimental data.

The main advantages of our approach are as follows. Our scheme is data-agnostic because it can consume samples from any source that provides data as a movement trajectory. The model does not necessarily assume a specific underlying theory and it could even be applied to scenarios for which no theory yet exists, i.e., it could be used to develop theory from data. The scheme is flexible and can be applied to any environment or scenario in which movement is needed, compared to other approaches, which employ a solitary global model that all agents will access and that maps movement to a single data-point in a library, so that only models that match the specific scenarios recorded in the data library can be simulated.

2 Related Work

Behavior-based ABMs represent the agency of walkers (usually pedestrians on streets) in simulation, using transition rules that describe how agents relate information that they encounter in simulation to their behavior (Benenson and Torrens 2004). Transition rules are often built using movement proxies such as random walks (Schweitzer 2003) or shortest-path planning (Nieuwenhuisen *et al.* 2007). In other cases, rules are sourced in actual behavioral mechanisms: occupancy behavior (Gipps and Marksjö 1985), steering (Reynolds 1999), vision (Turner and Penn 2002), space-time agendas (Paris *et al.* 2007), behavioral geography (Torrens 2007), collective behavior (Pelechano *et al.* 2008), or psychology (Allbeck *et al.* 2002).

Data-driven models of movement primarily use motion samples as input to procedural (rather than behavioral) heuristics. Motion planners (Witkin and Popović 1995), for example, use kinematic solvers to map motion capture data to synthetic character–rigs in simulation. In other cases, *trajectory* data (from Geographic Positioning Systems (GPS), video cameras, or radio triangulation) are used to estimate plausible space-time paths of movement. This can involve classifying trajectories into types of movement (Dodge *et al.* 2009; Makris and Ellis 2002; Johansson *et al.* 2008), or the development of heuristics to estimate likely next trajectories from given paths. Krumm & Horvitz (2007) and Alvarez-Garcia *et al.* (2010) introduced methods to achieve the latter for cars, for example. Lerner *et al.* (2007) introduced a trajectory-prediction model for *pedestrian* movement. Their scheme enabled searching through a corpus of trajectories (extracted from video footage) and copied a likely trajectory from that library to animated agent-characters in simulation, using similarity of real and simulated positioning relative to fixed infrastructure and velocity of ambient pedestrians as a mapping. Lee *et al*. (2007) introduced a similar scheme for group movement, using locally weighted learning to match agent movement to video-derived trajectory samples. Zhang *et al*. (2009) presented a statistical learning-based steering model to predict agents' velocity and direction from trajectory data, using a velocity-space (a set of quadratic curves fitted from trajectory data) to determine the candidate trajectory. Nguyen *et al*. (2005) introduced a hierarchical hidden Markov model to learn movement from trajectory data by treating internal and external factors that simulated agents' movement as hidden forcing factors for transition between movement types. Jankowski and colleagues have developed an innovative scheme for inferring movement *events* from trajectory data inherent in metadata associated with Online photograph repositories (Jankowski *et al.* 2010).

The trajectory-sampling approach promises to help in overcoming the problems of building individually-rich movement ABMs that match real-world behavior, but challenges remain. Data is a first challenge: data provided by positioning systems are more precise than those collected through observation (whether the observation is made by humans or pattern extraction from video footage) because they are immune to occlusions (Mezouar and Chaumette 2002) and shadowing (Nabbe *et al.* 2006). But, observation can provide context for position, relating it to the physical environment and human terrain in which movement occurred (Willis *et al.* 2004). In our approach, we avoid having to trade-off the two, by (1) using human-observed trajectory data, where trained observers are able to resolve obfuscating factors; (2) using on-screen GIS for data collection to provide guiding information that assists observers in tracking movement with positional accuracy; (3) using spatial analysis in post-processing to index data positions to a physical environment; and (4) using simulation to generate plausible ambient human terrain.

Scale presents a second challenge. Some of the trajectory-mining methods we discussed consider the macro-scale of movement (the trip); others consider the micro-scale (step-by-step). These scale differences also appear in behavioral representation. Some schemes treat high-level behaviors (i.e., those involving significant conscious thought) and others focus on lower-level (subconscious, autonomic) behavior. Really, all of these factors control human movement. In our approach, we continue a recent tradition (Pelechano *et al.* 2008), adopting a mixed scheme. We pay particular care to represent movement at its appropriate scale by (1) treating *low-level behavior* (sub-movements within a path) at the *micro-scale* of movement (the person, step-by-step) and (2) managing *higher-level behavior* as the conscious selection and assembly of lower-level behavior at a *macro-scale*.

The modeling approach or philosophy that is used can provide a third challenge, because it may lock a model-user into a particular way of framing her simulation, rather than flexibly supporting exploration. Many of the approaches we outlined are used in computer graphics (Thalmann and Musse 2007) and their design is *ad hoc* to those applications: built for movement in particular spaces, places, or contexts because they are required for repeatable (procedural) tasks. Realistic representation of behavior may not even be needed and trajectory-mining models are often applied as look-up tables, which cannot be generalized beyond the specific data-set used. We adopt a different approach by (1) learning (rather than simply mapping) behaviors from data, and (2) by codifying the produced knowledge into synthetic behaviors that *can* be ascribed to new places, spaces, and contexts.

3 Methodology

An overview of our approach is illustrated in Figure 1. We collected a range of trajectory data, which we stored in a Geographic Information System (GIS) and we used spatial analysis to calculate compound metrics that add value to data by describing multi-dimensions of walking. We developed a companion scheme for mining those data and building knowledge for individual agents in ABMs by learning the movement behavior of real walkers (children indoors and outdoors in school and adult pedestrians in outdoor urban settings) through data access, classification, clustering, weighting, and matching.

[Figure 1 goes here.]

3.1 Building Trajectory Samples

The first task is to build a library of trajectory samples that capture essential and distinct characteristics of walking movement. Realistically, *any* data-set that can be transformed into location trajectories could be used. We used data on real and synthetic (simulated) movement. We ran a six-month study in an American preschool in which trained human observers used GIS-based tracking software that we developed to run on tablet PCs to digitize children's (aged three to five) movement. The GIS provided a cartographic interface, upon which observers recorded the motion of walkers by sketching with a stylus (Figure 2); the system than interpreted the position and timing of those sketches, which was stored in a spatiotemporal database. Design of the interface was influenced by the observers' feedback during a month-long trial run, and was developed to maximize ease-of-use (labeling of common movement sources/sinks; hierarchical controls that mapped to common tasks) and to guide observers' hand-to-eye coordination when sketching, through representation of building floorplans and key infrastructure and grid-lines representing 1m^2 on-the-ground. Observers were restricted from interacting with the children and data were collected with participant approval and subject to the oversight of the Institutional Review Board for the protection of human subjects. (Additional, social interaction tags were also ascribed to walker paths by observers working in tandem, who annotated socio-behavioral interactions between participants. We do not make use of those data in this paper, but we are investigating how it might be used in further studies.) Data were collected during class schedule, each weekday, for six months.

[Figure 2 goes here.]

3.1.1 Building Compound Variables to Characterize Movement

The next step is to add value to those data, by determining essential compound characteristics of movement. These were determined as follows.

***Self-moving speed*:** The instantaneous moving speed (s_t) of a walker i at sampling point (location) p on a trajectory at time t_i was calculated as distance to the position at the previous time step $t_i - 1$ on a person's trajectory, qualified by the duration of the move.

$$s_{t_i} = \left\| p_{t_i} - p_{t_{i-1}} \right\|_2 / (t_i - t_{i-1}) \qquad \text{(i)}$$

***Self-moving direction*:** was calculated using a local coordinate system (Figure 3). A walker's moving direction α at sampling point p and time t_i was calculated as the offset angle from its current moving direction (which is the vector from its current position p_{t_i} to a next position $p_{t_i} + 1$ along the trajectory) to the local x-axis. By transforming to a local coordinate system (as opposed to a global coordinate system), agents can draw upon many more trajectory samples to train their behavior (Figure 4). Rather than a query returning one sample trajectory, the local system allowed for a *bundle* of samples to be returned, thereby better-informing the model.

[Figures 3 and 4 go here.]

Intended moving speed and direction are important characteristics because they can distinguish different movement behaviors among different individuals, even for conditions in which walkers might have the same *instantaneous* moving speed and direction. For tractability, we calculated the movement intention of walkers' speed (s_{avg}) and direction (d_{avg}) as the average of ten previous sampling points along a trajectory.

$$s_{avg,\,t_i} = \begin{cases} \sum_{j=0}^{i} s_{t_j}/i\,,\, i \le 10 \\ \sum_{j=i-10}^{i} s_{t_j}/i\,,\, i > 10 \end{cases} \tag{2}$$

$$d_{avg,t_i} = \begin{cases} \sum_{j=0}^{i} d_{t_j}/i\,,\, i \le 10 \\ \sum_{j=i-10}^{i} d_{t_j}/i\,,\, i > 10 \end{cases} \tag{3}$$

***Distance to target*:** We assumed that the end point (a destination or waypoint) of the trajectory of each walker was its moving target (the human observers in our study recorded data in this way), and we tagged this simply using Euclidean distance from a current sampling point to the moving target. (If the sink for a path was otherwise unknown, it could be implied by extended periods of motionless behavior.)

***Environmental variables*:** We accounted for walkers' interaction with the built environment using a detection filter to represent vision. This filter was a sector, centered on each agent and oriented in its direction of travel, with user-configurable radius and a 120 degree angle, i.e., average healthy field-of-vision (Cutting *et al.* 2000). All static objects that the filter passed over as an agent moved were recorded in the local coordinate system we described. Trading off computational efficiency and detail, we divided the filter into twelve equal subsectors (Figure 5), calculating distance to environmental obstacles per subsector, producing a (always unique, in our tests) twelve-dimensional vector. This vector could be interpreted as a signature for comparing similar data in the trajectory library. Because the signatures are expressed in a local coordinate system, a query to the library to return matching archived trajectory samples will produce many results. For example, in the scenarios depicted in Figure 4, samples 1 and 2 provide different built environments, but both sampling points have equivalent local relationships to their surroundings and the next actions along their trajectories are the same: moving towards free space without obstacles. This contrasts with global approaches [such as Lerner, et al (2007)], which only return a single sample per query because only the *specific* context of a particular environment is considered; outside of this context, that model may no longer be appropriate.

Human neighborhood variables**:** Walkers generally avoid collisions with other people in their vision and in a small zone of awareness around (including behind) themselves (Cutting *et al.* 1995). The latter zone usually produces rough awareness (because of aural and olfactory sense, memories of passing people, and cast shadows); the former is usually a reasonably accurate vector (Vishton and Cutting 1995). To represent this, we defined a circular zone for polling (human) neighborhood information, allowing its radius to be determined by the walker's instantaneous moving speed, as follows.

$$r = speed \times \beta + r_0 \tag{4}$$

Above, r denotes the detecting radius, β is a user-defined constant and r_0 is a user-defined minimum observation range for the walker. The circle is user-configurable to allow for specification of different detecting radii. In dense conditions, for example, the area may shrink to track only immediate neighbors (Vishton and Cutting 1995). For efficiency, we partitioned the circle in eight equal sectors, in which distance from an agent to its nearest-neighboring agent and the neighbor's speed and direction were recorded, producing an eight-dimensional vector.

[Figure 5 goes here.]

3.2 Data Control

The next step is to build a *low-level action* model that can use the compound vectors we just described to predict movement, and to construct a *higher-level behavioral* model to manage actions.

3.2.1 Action Control

The action controller delivers three autonomic behaviors (although others could be considered): (1) locomotion from an agent's last position toward a destination, while (2) instructing agents on how to avoid collisions with the physical environment, and (3) telling agents how to avoid collisions with nearby human agents. In both collision-avoidance schemes, agents determine their next action *per-encounter*, i.e., without a global algorithm provided *a priori*.

For example, in the simulations shown in Movie 1, at an initial stage ($t = 0$), each agent was assigned a start position, destination, and a *behavior* (an action, selected using a higher-level controller). Each agent used the locomotion model to predict its moving velocity and direction. At the same time, each agent scanned for and identified potential collisions with other agents and/or the physical environment. If a potential collision was detected, the agent *shifted its action* from using the locomotion model to using a collision avoidance model and thus adapted its moving velocity and direction to produce a collision-free path. Once the agent resolved the collision-avoidance maneuver, it reverted back to the locomotion model to control it motion through space and time toward the destination (Figure 1).

Shifts between qualitatively distinct movement actions are handled by a higher-level control scheme that first considers an agent's current moving condition C_q. The potential for transition to another action A_q (where q is the query time step) is determined as follows. (We considered locomotion and collision avoidance, but others could be added.)

$$A_{q,locomotion} \sim C_q\{M, E\} \tag{5}$$

$$A_{q,\text{collision avoidance}} \sim \mathrm{C_q}\{M, E, N\} \tag{6}$$

A_q is a two-dimensional vector: [*next moving velocity, next moving direction*] and C_q is a combination of multi-dimensional vectors. *M* is a five-dimensional vector describing an agent's motion variables: [*speed, direction, intended speed, intended direction, distance to target*]. *E* is a 12-dimensional vector representing an agent's environmental variables: [*distance to nearest obstacle in section 1 to 12 in a sector detection area*]. *N* is a 16-dimensional vector denoting an agent's human neighborhood information: [*distance to nearest agent in section 1 to 8 in a circular circle detection area, moving direction of agent detected in section 1 to 8*]. This produces a 17-dimension vector for locomotion and a 21-dimension vector for collision-avoidance, a 38-vector parameter space in total.

3.3 Learning from Data

Once compound vectors have been generated for trajectory samples, the procedure of learning from the data can begin. First, data are organized for fast querying, then mapped to an agent's query in the simulation, clustered as bundles of relevant data, trained, tuned, and delivered to the agent.

Collected data are classified into behavioral categories (e.g., walking, running), producing *N* action-condition samples $\{(A_i, C_i)\}$, where $i = 1,2, \ldots, N$. These data either come already tagged by behavior or the category is determined from velocity and speed. The task of finding relevant samples per agent query involves searching a small set of k training samples, $k \in \{(A_i, C_i)\}$, where $i = 1,2,\ldots,K$, are those most similar to query point C_q (an agent's instantaneous moving condition).

Finding similar samples is difficult, especially when the dimensionality of query points is high. Tan *et al.* (2006) showed that the computation time for matching can be improved with approximations of nearest-neighbor samples, with relatively small error. We achieved this using the *K-Nearest Neighbor* (KNN) algorithm (Arya *et al.* 1998) to find *k* most similar samples from the trajectory library (we interfaced Mount and Arya's (1997) ANN library in our code). KNN organizes samples into a *K-Dimensional* (*k-d*) tree in memory (Yao 1977), where *k* denotes the dimension of vector space for each sample. We stored different types of samples in different *k-d* trees. (For this study the *k* values of the *k-d* tree were 17 dimensions for locomotion samples and 21 dimensions for collision avoidance samples.) Once organized, k-nearest neighbors may be sought from the library, based on the distance from a query point C_q to a sample C_i, where $i = 1,2,\ldots,N$. We used Euclidean distance between two vector spaces in a two-norm form:

$$d_E(x,q) = \|x - q\|_2 \qquad (7)$$

Above, x and q are two vector spaces of sample point C_i and query point C_q.

Each agent uses KNN to search for k-nearest, most similar (to its current status), samples and these samples are then used to calibrate either a locomotion or a collision avoidance model that will enable prediction of the agent's future movement action A_q. Once the agent learns its moving action and moves to a new position, it re-queries nearest samples and calibrates a new model again to predict its next movement. To facilitate regression and to restrict the number of overall samples in the *k-d* tree, the value of k should be larger than the dimension of the vector space of condition variables C_q; we allowed k to be user- and scenario-configurable.

3.3.1 Choosing Clusters as Bundles of Relevant Trajectory Samples

Results of agent queries are non-Markovian, so k researched samples might contain distinct clusters representing completely different movement actions, e.g., when approaching the middle of a wall, people may turn right or left to avoid the wall with equivalent reaction (Figure 6). To resolve equivalencies in simulation, we apply the *K-Means Clustering* (KMC) algorithm (Kanungo *et al.* 2002) as selection logic. KMC allows us to find groups of objects that are similar or related to one another, but also different from (or unrelated to) objects in other groups. This is achieved by associating clusters with a centroid point, so that each point may be assigned to the cluster with the closest centroid. (The number of clusters *K* must be specified; we defined $K = 3$.) In our work, we first calculated k searched samples using KNN and we then applied KMC to every set *k* to produce clusters. The reasonability of a given cluster (i) was assessed as a possibility p_i of selecting i, given m_i, the mean value of output values in cluster *i*; A_q, the mean value of output action of the previous query point; and N_i, the number of samples in cluster *i*, where *N* is the total number of samples *k* searched by KNN.

$$p_i = exp\,(-\,(m_i - \,A_q)^2 N_i/N) \tag{8}$$

p_i has possible positive correlation with the deviation of previous queried samples and the number of samples in each one of $K = 3$ clusters, so we removed the cluster with smallest *p* value from consideration first. We applied a random selection for the remaining two *p* values if walkers were very close to each other, to represent stochasticity in movement behavior. We employed a configurable threshold to determine when to apply randomness; otherwise, the cluster with the highest *p* value was be chosen.

[Figure 6 goes here.]

3.3.2 Locally Weighted Regression

After we choose appropriate movement action samples from a trajectory database, we can use samples to calibrate simulated locomotion and collision avoidance models to real data. We performed calibration using locally weighted regression (LWR) (Atkeson *et al.* 1997), in conjunction with KNN and KMC. LWR enables *local fitting*. Unlike traditional regression, for which a single global model is estimated, LWR consumes training data only in a *nearby* region around the location of a query point when fitting a prediction surface, working on an assumption that closer samples have more impact on movement than those at-a-distance. We applied Euclidean distance between a query point and any nearby point, with a Gaussian kernel function, $0 \leq K(d) \leq 1$ to weight the distance d between a sample point x and a query point q. Alternative functions could be used.

$$K(d) = \exp(-d^2) \tag{9}$$

For our applications, we wished to estimate an agent's moving speed and direction per situation, so we needed to fit two different LWR models, as follows.

$$y_{i,speed} = \alpha_0 + \sum_k \alpha_k x_{ik} + \varepsilon_i \tag{10}$$

$$y_{i,direction} = \beta_0 + \sum_k \beta_k x_{ik} + \varepsilon_i' \tag{11}$$

Above, $y_{i,speed}$ and $y_{i,direction}$ are two outputs (speed and direction) of agent i. α_0 and β_0 are intercepts of the regression line. x_{ik} is the k^{th} element of the k-dimensional explanatory variables, i.e., the moving condition variables defined in formulae (5) and (6). ε_i and ε_i' are well-behaved disturbance error terms with zero-mean and constant variance. α_k and β_k are the regression coefficients we needed to estimate from training data.

In matrix notation, the unknown regression coefficients vectors α and β of the two LWR models could be estimated as follows.

$$\alpha = (X^T \mathrm{W} \mathrm{X})^{-1} X^T \mathrm{W} y_{speed} \tag{12}$$

$$\beta = (X^T \mathrm{W} \mathrm{X})^{-1} X^T \mathrm{W} y_{direction} \tag{13}$$

Above, X is a matrix in which each row is a moving condition vector x_i. W is a diagonal weighting matrix in which each diagonal element w_{ii} is calculated using a Gaussian decay function.

$$\mathrm{w}_{ii} = \; exp\,(-(d/h)^2) \tag{14}$$

Above, *d* represents the Euclidean distance between input sample points and a query point and *h* is a bandwidth that defines the scale or range over input samples. For computational efficiency, we applied the nearest neighbor bandwidth selection that *h* is equal to the distance from query point to the k^{th} nearest training sample point. This causes the input data volume to change according to the density of searched nearby samples.

3.3.3 Tuning

LWR may fail to return acceptable output if training samples are not well-selected (e.g., when the situations encountered by agents in simulation are significantly different from any real-world samples or they are completely novel, with no analog in the data), so an assessment of fit is needed. However, it is costly to estimate goodness-of-fit of our LWR models because they *run in real-time*, i.e., LWR is applied at every simulation step of every agent. Each LWR procedure uses KNN and KMC and the computational time required by LWR depends heavily on the size of all sample spaces and the length of independent vectors. For example, if we simulated the movement of ten agents with thirty frames per second resolution, LWR would need to be executed $1{,}000(30) = 30{,}000$ times per second.

As an alternative to estimating prediction error, we defined a simple rule to tune excessive or unacceptable output of the models. First, in each LWR procedure, after using KMC to identify the most reasonable clusters, we calculated statistical indicators from the samples in the selected cluster: minimum, average, and maximum value of moving speed and moving direction. Second, we defined a criterion to determine if the predicted outputs were acceptable or not by checking if the output moving speed and direction fell in the range of minimum and maximum moving speed and direction, which we calculated in the first step. When the output values exceeded the acceptable value range, we tuned the outputs by using the average moving speed and direction as a substitute for the unacceptable outputs.

3.4 Assembling Behaviors as Finite State Machines

Atop the scheme we have described, we designed a high-level Finite State Machine (FSM) controller, as a *scenario-based configuration model*. In the FSM, each learning-based low-level action model is treated as a single state and the FSM determines transitions between states, as a higher-level controller. This allowed us to endow simulation agents with the ability to mix-and-match different lower-level behaviors, to account for *different behavioral faculties* as different durations and transition potentials of the low-level action model.

Take a simple scenario as an example: if users wished to explore or reproduce an evacuation scenario, they could first build three low-level action models, which could be learnt from collected trajectories of realistic walkers. Suppose that three learning-based models were provided—walk, run, and stand—and all agents had the same movement behavior. Then, users could define a FSM for all agents (Figure 7). This FSM might dictate that each agent will start with a running action in a panic situation, and end with a standing action when they reach a safe destination for assembly. Agents' activity state transition—from running to walking—could take place when they successfully escape. Users could also define a state transition from running to standing when agents get stuck in a crowd, or assign symbolic waypoints to agents (a start position, an exit waypoint, and a safe assembly position, for example). Other control models could be implemented per application, event, demographic, place, etc.

[Figure 7 goes here.]

3.5 Analyzing Simulated Movement

We calculated metrics of movement to compare and contrast real-world and simulated paths and to assess differences between simulation scenarios. We calculated the following characteristics of movement: relative sinuosity as an indicator of perturbation [pedestrians prefer to move in relatively straight lines (Hillier and Hanson 1984)], scaling as an indicator of the spatial hierarchy of action-reaction, and trajectory preservation as an indicator of the model's ability to generate realistic movement at micro-scales.

Fractal dimension measures of the ability of a path to fill space. This can be interpreted in two important ways. First, it indicates the level of behavior: agents that move predominantly by low-level behavior (automotive locomotion) may generate paths with $\bar{D} \sim 1$, i.e., close to a straight line. Walkers that use collision avoidance may produce paths that fill more space, i.e., $1 < \bar{D} < 2$. Second, differences in the value can be used to measure relative sinuosity between agents, environments, models, sample-sets, scenarios, etc. We estimated fractal dimension D using traditional divider methods (Mandelbrot 1977). Mean fractal dimension ($\bar{D}$) was used to correct for known overestimation errors in truncation when using the divider method at large scales (Nams 2006).

Mean cosine of turning angle ($\overline{\cos}\,\theta$) indexes relative straightness over an *entire path* ($\overline{\cos}\,\theta = 1$ is straight) by relating total path length (*Net*) with step lengths (*step*) for different-sized dividers (Nams 2006).

$$\frac{\log(2)}{\log\left(\frac{Net}{step}\right)} = \frac{\theta}{1+\log_2(\cos\theta+1)} \qquad (15)$$

Correlation between successive turn angles ($corr\,\theta$) measures relative directional preservation on a *step-by-step* basis (Nams 1996).

Approximate Entropy (ApEn) indicates the likelihood of similar patterns to manifest in a time series (Pincus 1991). ApEn will be relatively small for data that contain many repetitive patterns (data that are highly structural) and high for data with a less predictable process (data with complex or random structure). Given data S_N with N continuous observations, we defined a sequence of m observations at location i, $i \in [1, N]$, as a pattern $p_m(i)$. If the difference between two patterns—$p_m(i)$ and $p_m(j)$—is less than a predefined criterion b, we may regard these patterns as similar.

$$ApEn(S_N, m, b) = \; ln\left[\frac{C_m(b)}{C_{m+1}(b)}\right] \qquad (16)$$

Above, m specifies the pattern length, b defines the criterion of similarity between patterns, and $C_m(b)$ is the prevalence of repetitive patterns of length m in S_N, which can be calculated as follows.

$$C_m(b) = \sum_{i=1}^{N-m+1} n_{im}(b) \Big/ (N-m+1)^2 \qquad (17)$$

Above, $n_{im}(b)$ is the number of patterns in P_m that are similar to $p_m(i)$.

4 Experiments

We developed several standalone, machine-learned, simulations to prove the usefulness of our model using different scenarios and data. (Parameterization of the scenarios is detailed in Table 1.) We also compared machine-learned movement to real-world movement, using the preschool data we already discussed, as well as GPS trajectories that we obtained for adult pedestrians in downtown Salt Lake City, USA and Yokohama, Japan.

[Table 1 goes here.]

4.1 Preschool Children's Movement Behavior

In this experiment, we trained the model on the preschool data described in section 3.1. 89,518 samples were used for walking or running and 70,527 for cycling (Figure 8). Arbitrary source and sink locations in a simulated replication of the preschool environment were provided and the model had to learn how to move free from collisions between these locations, using only the training data. The resulting simulation is shown in Figure 9 and Movie 1. Visually, machine-learned child-walking appeared appropriately child-like with some unneeded sinuosity, which is understandable, given that the model is sourced from data of mobile toddlers. In the cycling scenario, machine-learning child-bikers managed to move along a track, successfully turning corners while staying within track boundaries with visually-plausible movement for small tricycles with a top speed of ~1 mile per hour.

[Figures 8 and 9 go here.]

Values of fractal mean ($\overline{D}$) for real-world and machine-learned kids' walking and tricycling were a good match: $\overline{D}$~1.02 for observed tricycling and $\overline{D} = 1.01$ for machine-learned tricycling; $\overline{D} = 1.02$ to 1.16 for observed walking and $\overline{D} = 1.02$ for machine-learned walking (Table 2). These values imply that movement occupied a relatively small amount of space and was therefore low in sinuosity and close to straight, considered over the *entire path*. However, machine-learned kids' movement was straighter than that of real-world observations at *local* scales ($\overline{\cos}\,\theta = 0.98$ for machine-learned tricycling, compared to 0.56 to 0.79 for observed tricycling; $\overline{\cos}\,\theta = 0.91$ for machine-learned walking, compared to 0.64 to 0.82 for observed walking). A greater degree of directional preservation was reported for machine-learned movement, step-by-step ($corr\,\theta = 0.01$ for machine-learned biking and -0.02 for machine-learned walking, compared to -0.63 to -0.39 for observed biking and -0.64 to -0.47 for observed walking). Again, this indicates that machine-learned movement was *locally* straighter than in the real world.

[Table 2 goes here.]

Considering that toddlers' playing is often target-less, random, and disordered, the ApEn measurements of children's movement in preschool should be high, i.e., movement should be unpredictable. This was evident in our measurements (Table 3): the average ApEn values of real children's walking velocity, acceleration, and turning angles were about 0.61, 0.62 and 0.73, respectively, close in value to ApEn for machine-learned walking, which were 0.65, 0.82, and 0.71, respectively. By these measures, our machine-learned walking model produced plausibly complex behavior. The average ApEn values for machine-learned cycling velocity, acceleration, and turning angles were 0.97, 1.13, and 1.03, respectively, higher than the average ApEn values for real children's cycling, which were 0.52, 0.56, and 0.51, respectively (Table 4). This implies that our machine-learned model *over-estimated* the amount of complexity required to produce movement by tricycling.

[Table 4 goes here.]

All of these values were considerably higher than the same ApEn measurements of real pedestrian walking in Salt Lake City (0.18 for velocity, 0.14 for acceleration, and 0.18 for turning). Adult pedestrians can feasibly be considered as walking with greater skill and purpose in cities than toddlers in a preschool might do. Because adults have the ability to control their moving speed and turning angles to move smoothly and comfortably towards their target, they will usually change their moving velocity, acceleration, and turning angles progressively, which will persist more continuously across a trip. In other words, they are more predictable than toddlers.

4.2 Meta-Simulating Movement While Playing Capture-the-Flag

This simulation tasked modeled agents with the goal of capturing a stationary flag in a simulated space, using machine-learned behavior to move as quickly as possible while avoiding collisions with fixed objects. In this case, we used machine-learning as a *meta*-ABM. First, we simulated movement among agents in a crowded environment using Reynolds's (1999) steering model (Movie 1). Second, we used 11,000 trajectory samples from this simulation to learn *new* movement between arbitrary (non-sampled) origins and destinations in a *second* agent-based simulation that relied only on our learning scheme (i.e., the second model did not have access to Reynolds's algorithms at all). Because the focus was on avoiding fixed obstacles only, one agent was introduced to the simulation at a time. The machine-learning routine did produce visually smooth, collision-free movement for these scenarios (Movie 1). We also tasked the machine-learning model with producing movement that was free from collisions with fixed (infrastructure) obstacles *and* with mobile (other agent) obstacles (Movie 1).

Values of fractal metrics were close to those for the original simulated Reynolds steering paths, as were metrics for ApEn velocity, acceleration, and direction. Results for the machine-learned meta-ABM reflected the heuristic of the Reynolds algorithms, which was to move agents to destinations by smoothly avoiding collisions. Agents' velocity change was implemented by applying a braking or accelerating function, which yielded a fully "predictable" pattern. This was apparent in our ApEn results: movement by machine-learned capture-the-flag behavior was a plausible match to the original steering data produced by the Reynolds simulation (Table 4).

4.3 Quotidian Walking along Downtown Streetscapes

In this experiment, we introduced *two*—qualitatively different—sets of training data, generated using Reynolds (1999) steering algorithms: (1) a game of capture-the-flag in an obstacle-littered environment (fixed obstacle avoidance) and (2) agent-agent avoidance in free space (mobile collision avoidance) (Movie 1). These data were used to feed a machine-learning model of everyday walking behavior along a simulated streetscape (our scheme allows simulated environments to be specified using ASCII, bitmap, or shapefiles; agents can interpret boundaries in any of these formats).

The idea, in this scenario, was to test the ability of the FSM controller to switch between qualitatively-different movement behaviors (walking, stopping, steering to avoid collisions, avoiding fixed and mobile obstacles). Agents did move in a visually realistic way to their destinations, employing sequences of walking, steering, and collision avoidance to get out of the way of fixed and mobile obstacles (Figure 10).

[Figure 10 goes here.]

We compared the resulting machine-learned simulation to movement paths for real-world (adult) urban pedestrians (Figures 11 and 12). Fractal values for the American ($\overline{D} = 1.04$) and Japanese ($\overline{D} = 1.16$ to 1.21) samples differed, reflecting the relatively denser pedestrian streetscapes of Yokohama, in which pedestrians must execute a greater number of collision-avoidance maneuvers to avoid bumping into other people. Differences also presented in values for directional preservation: the Japanese examples were significantly less straight over an entire trip ($\overline{\cos}\,\theta = 0.36$ to 0.46) than the American example ($\overline{\cos}\,\theta = 0.92$). $corr\,\theta$ was strongly negative for the urban samples, suggesting a lack of directional preservation. The machine-learned urban pedestrian model generated measures of fractal mean ($\overline{D} = 1.02$) and directional preservation ($\overline{\cos}\,\theta = 0.99$) that corresponded well with the American movement case, but the fractal measure was significantly less sinuous and the directional preservation was straighter than the Japanese samples. This could, perhaps, be explained by the lower number of potential collisions in the ambient pedestrian traffic in that simulation, compared to the reality of downtown Japanese streetscapes. For the same reason, we might reasonably expect pedestrian movement in the Yokohama case to be more complex and unpredictable than the American case. Indeed, we measured relatively high ApEn values for Yokohama moving velocity and acceleration: 0.94 in velocity and 0.59 in acceleration (see Table 3). The ApEn measurement of direction was only 0.16, which could be attributed to the tendency for dense crowds to self-organize into spontaneous lanes of unidirectional movement (Helbing *et al.* 2001). Machine-learned urban pedestrian movement produced ApEn velocity and acceleration values that fell roughly halfway between those for real-world walking in Salt Lake City and Yokohama (Table 4), although ApEn direction values were a very close match for machine-learned and real-world urban movement.

[Figures 11 and 12 go here.]

5 Conclusions

In this paper, we introduced a novel method for machine-learning human movement behavior, on-the-fly, from large databases of trajectory samples. It is important to note that this is distinct from (1) simply reproducing movement *patterns* in simulation and from (2) using a model *a priori*. Using a series of experiments to simulate real-world scenarios, we have shown that the scheme can be used to automatically generate realistic-looking and quantitatively plausible movement behavior for non-sampled situations and that different types of trajectory data can be inter-mingled in the scheme, as needed. The approach that we introduced is novel relative to other machine-learning schemes for movement, because it is data-agnostic and because it can be applied to different environments without the need to be reconfigured anew. This novelty is achieved by modeling, weighting, training, and applying space-time specific models per agent, per location, and per time-step in simulation, which contrasts with standard approaches that use global models that may work only on particular data-sets and for particular scenarios.

There are other potential applications of the work. We focused on human movement, but the scheme could be applied to animal or insect movement [indeed, data for these contexts are often more readily available (Nathan *et al.* 2008)]. Our model could be adapted to work on real-time data-feeds, such as those generated from location-based services. The FSM controller that we introduced, while straightforward, could be expanded to encompass more complicated (agent-based, or other schemes) models, or to dock to existing models that need not necessarily be movement-based. Indeed, our scheme could be used to spatially enable non-mobile models. The scheme could also be used for what-if experimentation in agent-based models with hypotheses that are informed or translated from data and the construction of algorithms for extracting and annotating space-time paths in massive data-sets. This potentially allies our scheme with developing functionality in computational social science (Lazer *et al.* 2009; Torrens 2010b).

The approach, as presented, does have some deficiencies and room for improvement remains. First, our model depends heavily on trajectory samples. Because learning is data-driven, action models can only be reliably learned from representative samples. This means, for example, that ordinary data (e.g., commuting) could not be used to train a model for extraordinary movement (e.g., panicked stampedes); a new model would need to be developed for these behaviors. The precision of the machine-learning is also influenced by collected trajectory data and different data sources (observation, GPS, video sampling) yield different advantages and disadvantages; no one technique is both automatic and precise. Intuitively, combing several different data sources would provide a solution to this problem. Second, we have abstracted from discussion of the computational cost of our scheme. Memory is not a problem: we employed a *k-d* tree to efficiently index huge sample datasets. However, requirements for KNN querying and regression-based learning will increase with the amount of training data infused to the algorithms at run time.

Development of this scheme is ongoing and in the future, we plan to extend the approach to other movement behaviors, covering different demographics of agents and moving conditions and to accommodate near-real-time data feeds. We also plan to introduce higher-level movements, beyond local choreography to include trip-planning, path-planning, and way-finding. We are also investigating methods for accelerating the computation of the model using coarse-grained multi-scale computation (Kevrekidis *et al.* 2003).

Acknowledgements

This material is based in part upon work supported by the National Science Foundation under grant numbers 1002517, 0643322, and 0624208. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

Allbeck, J., K. Kipper, C. Adams, W. Schuler, E. Zoubanova, N. Badler, M. Palmer, and A. Joshi. 2002. ACUMEN: amplifying control and understanding of multiple entities. Proceedings of The First International Joint Conference on Autonomous Agents and Multiagent Systems, 191-198, at Bologna, Italy.

Alvarez-Garcia, J. A., J. A. Ortega, L. Gonzalez-Abril, and F. Velasco. 2010. Trip destination prediction based on past GPS log using a Hidden Markov Model. *Expert Systems with Applications* In Press.

Arya, S., D. Mount, N. Netanyahu, R. Silverman, and A. Wu. 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)* 45 (6):891-923.

Atkeson, C. G., A. W. Moore, and S. Schall. 1997. Locally weighted learning. *Artificial Intelligence Review* 11 (1):11-73.

Benenson, I., and P. M. Torrens. 2004. *Geosimulation: Automata-Based Modeling of Urban Phenomena*. London: John Wiley & Sons.

Cutting, J. E., P. M. Z. Alliprandini, and R. F. Wang. 2000. Seeking one's heading through eye movements. *Psychnomic Bulletin & Review* 7 (3):490-498.

Cutting, J. E., P. M. Vishton, and P. A. Braren. 1995. How we avoid collisions with staionary and moving obstacles. *Psychological Review* 102 (4):627-651.

Dodge, S., R. Weibel, and E. Forootan. 2009. Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. *Computers, Environment and Urban Systems* 33 (6):419-434.

Gibson, W. 1984. *Neuromancer*. New York: Ace Books.

Gipps, P. G., and B. Marksjö. 1985. A microsimulation model for pedestrian flows. *Mathemathics and Computers in Simulation* 27:95-105.

Helbing, D., P. Molnár, I. Farkas, and K. Bolay. 2001. Self-organizing pedestrian movement. *Environment and Planning B* 28:361-383.

Hillier, B., and J. Hanson. 1984. *The Social Logic of Space*. Cambridge: Cambridge University Press.

Hughes, R. L. 2003. The flow of human crowds. *Annual Review of Fluid Mechanics* 35:169-182.

Jankowski, P., N. Andrienko, G. Andrienko, and S. Kisilevich. 2010. Discovering Landmark Preferences and Movement Patterns from Photo Postings. *Transactions in GIS* 14 (6):833-852.

Johansson, A., D. Helbing, H. Al-Abideen, and S. Al-Bosta. 2008. From crowd dynamics to crowd safety: a video-based analysis. *Advances in Complex Systems* 11 (4):497-527.

Kanungo, T., D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. 2002. An efficient means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (7):881-892.

Kevrekidis, I. G., C. W. Gear, J. M. Hyman, P. G. Kevrekidid, O. Runborg, and C. Theodoropoulos. 2003. Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis. *Communications in Mathematical Sciences* 1 (4):715-762.

Krumm, J., and E. Horvitz. 2007. Predestination: Where do you want to go today? *IEEE Computer* 40 (4):105-107.

Lazer, D., A. Pentland, L. Adamic, S. Aral, A.-L. Barabási, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, T. Jebara, G. King, M. Macy, D. Roy, and M. Van Alstyne. 2009. Computational social science. *Science* 323 (5915):721-723.

Lee, K. H., M. G. Choi, Q. Hong, and J. Lee. 2007. Group behavior from video: a data-driven approach to crowd simulation. Paper presented at 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, eds. M. Gleicher and D. Thalmann, at San Diego, California, August 3-4.

Lerner, A., Y. Chrysanthou, and D. Lischinski. 2007. Crowds by Example. *Computer Graphics Forum* 26:655-664.

Makris, D., and T. Ellis. 2002. Path detection in video surveillance. *Image and Vision Computing* 20 (12):895-903.

Mandelbrot, B. 1977. *The Fractal Geometry of Nature*. San Francisco: W.H. Freeman.

Mezouar, Y., and F. Chaumette. 2002. Avoiding self-occlusions and preserving visibility by path planning in the image. *Robotics and Autonomous Systems* 41 (2-3):77-87.

Mount, D., and S. Arya. 1997. ANN: A library for approximate nearest neighbor searching.

Nabbe, B., D. Hoeim, A. A. Efros, and M. I. Herbert. 2006. Opportunistic use of vision to push back the path-planning horizon. Proceedings of 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, ed. Y. Liu, 2388-2393, at Beijing, October 9-15.

Nams, V. O. 1996. The VFractal: a new estimator for fractal dimension of animal movement paths. *Landscape Ecology* 11 (5):289-297.

———. 2006. Improving accuracy and precision in estimating fractal dimension of animal movement paths. *Acta Biotheoretica* 54 (1):1-11.

Nathan, R., W. M. Getz, E. Revilla, M. Holyoak, R. Kadmon, D. Saltz, and P. E. Smouse. 2008. A movement ecology paradigm for unifying organismal movement research. *Proceedings of the National Academy of Sciences* 105 (49):19052-19059.

Nguyen, N. T., D. Q. Phung, S. Venkatesh, and H. Bui. 2005. Learning and detecting activities from movement trajectories using the hierarchical hidden markov models. Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), eds. M. Hebert and D. Kriegman, 955-960, at San Diego, CA, June 20-26.

Nieuwenhuisen, D., A. Kamphuis, and M. H. Overmars. 2007. High quality navigation in computer games. *Science of Computer Programming* 67 (1):91-104.

Paris, S., J. Pettré, and S. Donikian. 2007. Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum* 26 (3):665-674.

Pelechano, N., J. Allbeck, and N. I. Badler. 2008. *Virtual Crowds: Methods, Simulation, and Control*. San Rafael, CA: Morgan & Claypool.

Pincus, S. M. 1991. Approximate entropy as a measure of system complexity. *Proceedings of the National Academy of Sciences* 88 (6):2297-2301.

Reynolds, C. W. 1999. Steering behaviors for autonomous characters. In *Proceedings of the Game Developers Conference, 1999*, ed. Game Developers Conference, 763-782. San Jose, CA: Miller Freeman Game Group.

Schwartz, M. S., and C. G. Schwartz. 1955. Problems in participant observation. *The American Journal of Sociology* 60 (4):343-353

Schweitzer, F. 2003. *Brownian Agents and Active Particles*. Berlin: Springer-Verlag.

Tan, P., M. Steinbach, and V. Kumar. 2006. *Introduction to data mining*: Pearson Addison Wesley Boston.

Thalmann, D., and S. R. Musse. 2007. *Crowd Simulation*. London: Springer-Verlag.

Torrens, P. M. 2007. Behavioral intelligence for geospatial agents in urban environments. Proceedings of IEEE Intelligent Agent Technology (IAT 2007), eds. T. Y. Lin, J. M. Bradshaw, M. Klusch and C. Zhang, 63-66, at Los Alamitos, CA.

———. 2010a. Agent-based modeling and the spatial sciences. *Geography Compass* 4 (5):428-448.

———. 2010b. Geography and computational social science. *GeoJournal* 75 (2):133-148.

Turner, A., and A. Penn. 2002. Encoding natural movement as an agent-based system: an investigation into human pedestrian behaviour in the built environment. *Environment and Planning B: Planning and Design* 29 (4):473 - 490.

Vishton, P. M., and J. E. Cutting. 1995. Wayfinding, displacements, and mental maps: velocity fields are not typically used to determine one's aimpoint. *Journal of Experimental Psychology* 21 (5):978-995.

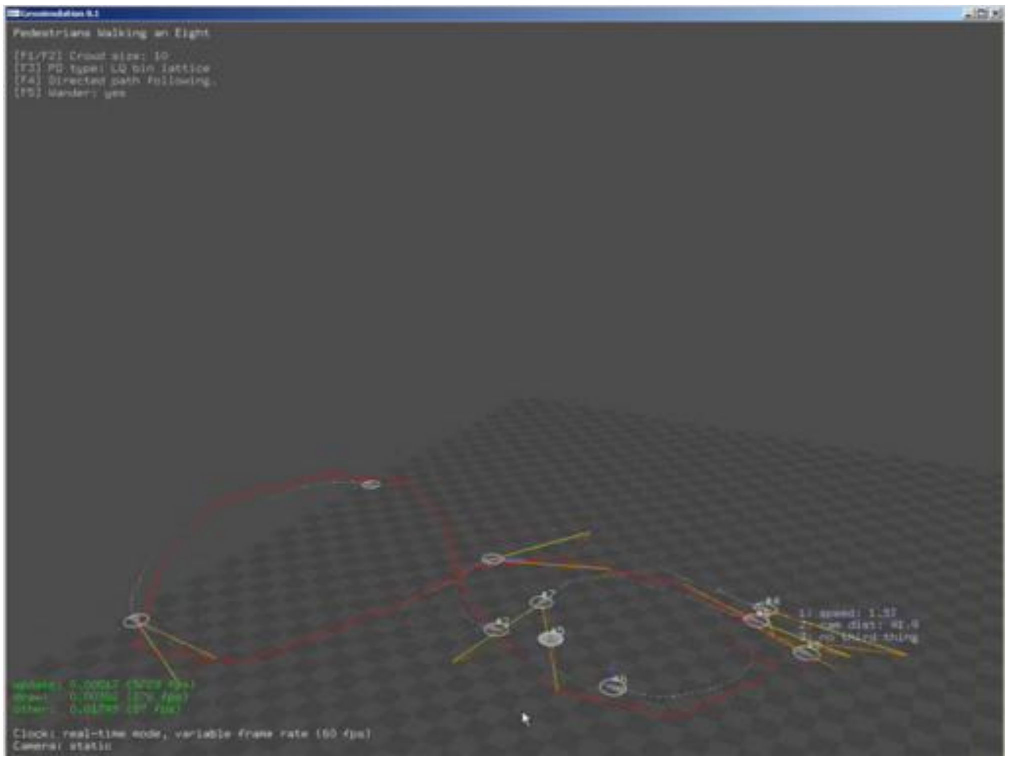
Willis, A., N. Gjersoe, C. Havard, J. Kerridge, and R. Kukla. 2004. Human movement behaviour in urban spaces: implications for the design and modelling of effective pedestrian environments. *Environment and Planning B* 31 (6):805-828.

Witkin, A., and Z. Popović. 1995. Motion warping. Proceedings of 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), ed. M. Cohen, 105-108, at Los Angeles, CA, August 6 to 11.

Yao, A. C. 1977. *On constructing minimum spanning trees in k-dimensional spaces and related problems*. Palo Alto, CA: Stanford University, Department of Computer Science. Technical Report CS-TR-77-642.

Zhang, Y., J. Pettré, Q. Peng, and S. Donikian. 2009. Data based steering of virtual human using a velocity-space approach. In *Proceedings of the Second International Workshop on Motion in Games (Lecture Notes in Computer Science 5884)*, eds. A. Egges, R. Geraerts and M. Overmars, 170-181. Zeist, The Netherlands: Springer-Verlag.

Movies



```
Pedestrians Walking an Eight

[F1/F2] Crowd size: 10
[F3] PD type: LQ bin lattice
[F4] Directed path following.
[F5] Wander: yes

1: speed: 1.37
2: cam dist: 41.6
3: no third thing

Clock: real-time mode, variable frame rate (60 fps)
Camera: static
```

"*TGIS-movie.wmv*" is attached with the submission of the manuscript; it shows the simulations that the machine-learning scheme is capable of generating.

Figures

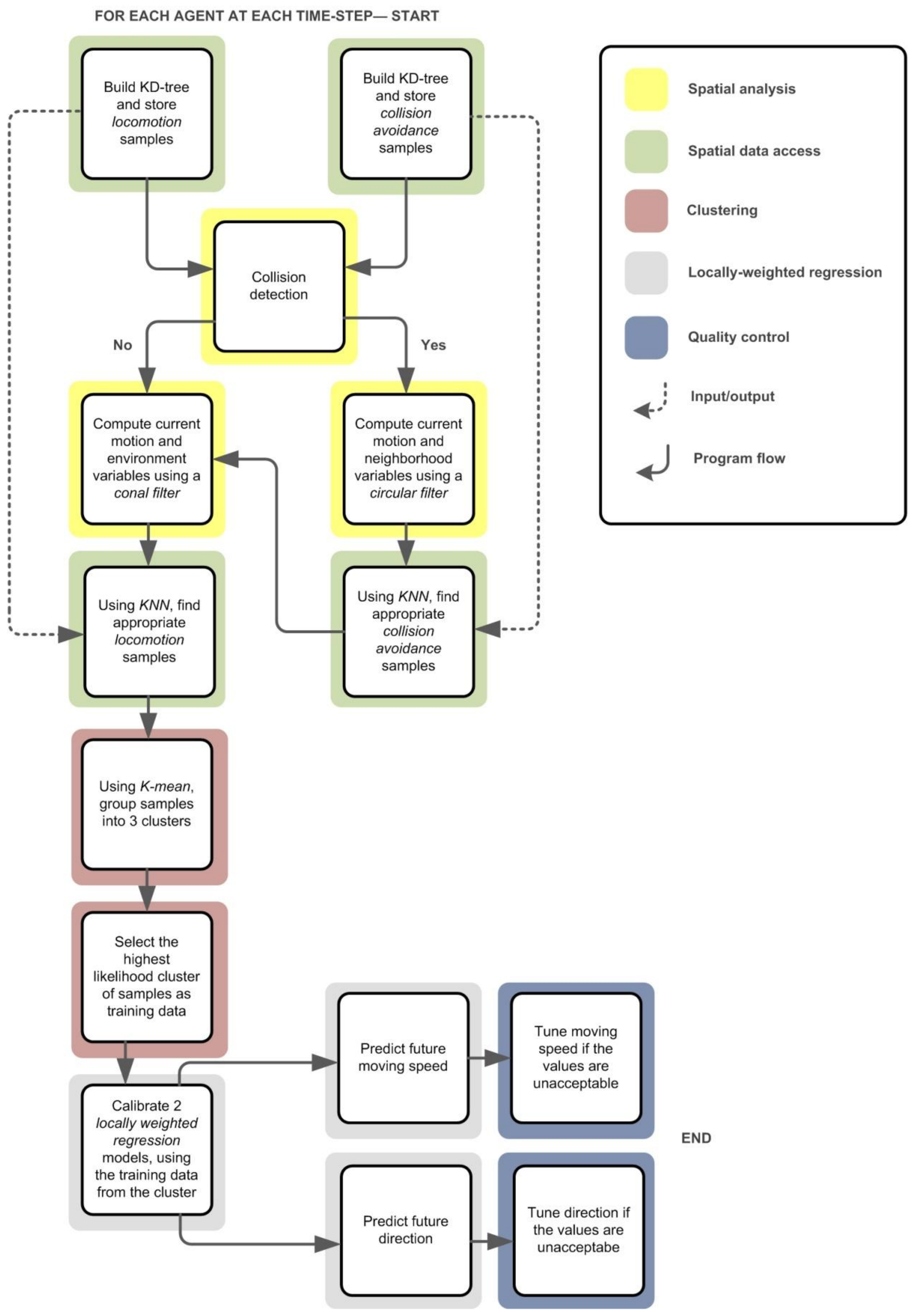

FOR EACH AGENT AT EACH TIME-STEP— START
Build KD-tree and store *locomotion* samples
Build KD-tree and store *collision avoidance* samples
Collision detection
No
Yes
Compute current motion and environment variables using a *conal filter*
Compute current motion and neighborhood variables using a *circular filter*
Using *KNN*, find appropriate *locomotion* samples
Using *KNN*, find appropriate *collision avoidance* samples
Using *K-mean*, group samples into 3 clusters
Select the highest likelihood cluster of samples as training data
Calibrate 2 *locally weighted regression* models, using the training data from the cluster
Predict future moving speed
Tune moving speed if the values are unacceptable
Predict future direction
Tune direction if the values are unacceptabe
END
Spatial analysis
Spatial data access
Clustering
Locally-weighted regression
Quality control
Input/output
Program flow

Figure 1. The programmatic flow between different components of the *low-level action model*.

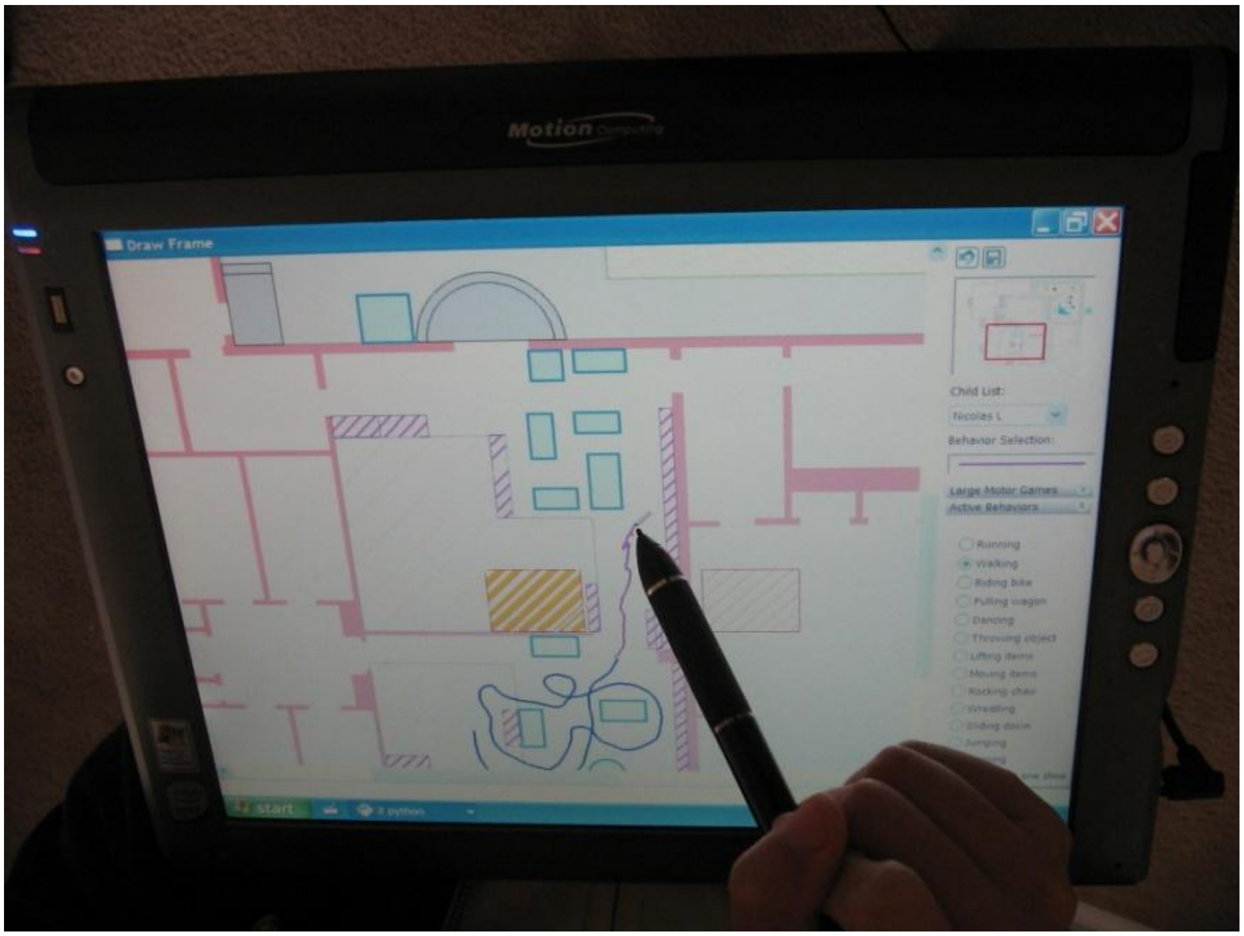


Figure 2. Trajectory samples of children's movement were collected by trained human observers, who sketched their movement in real-time, using a GIS-based GUI on a tablet PC.

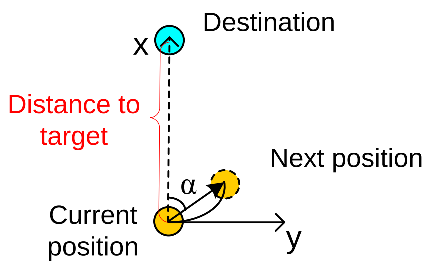
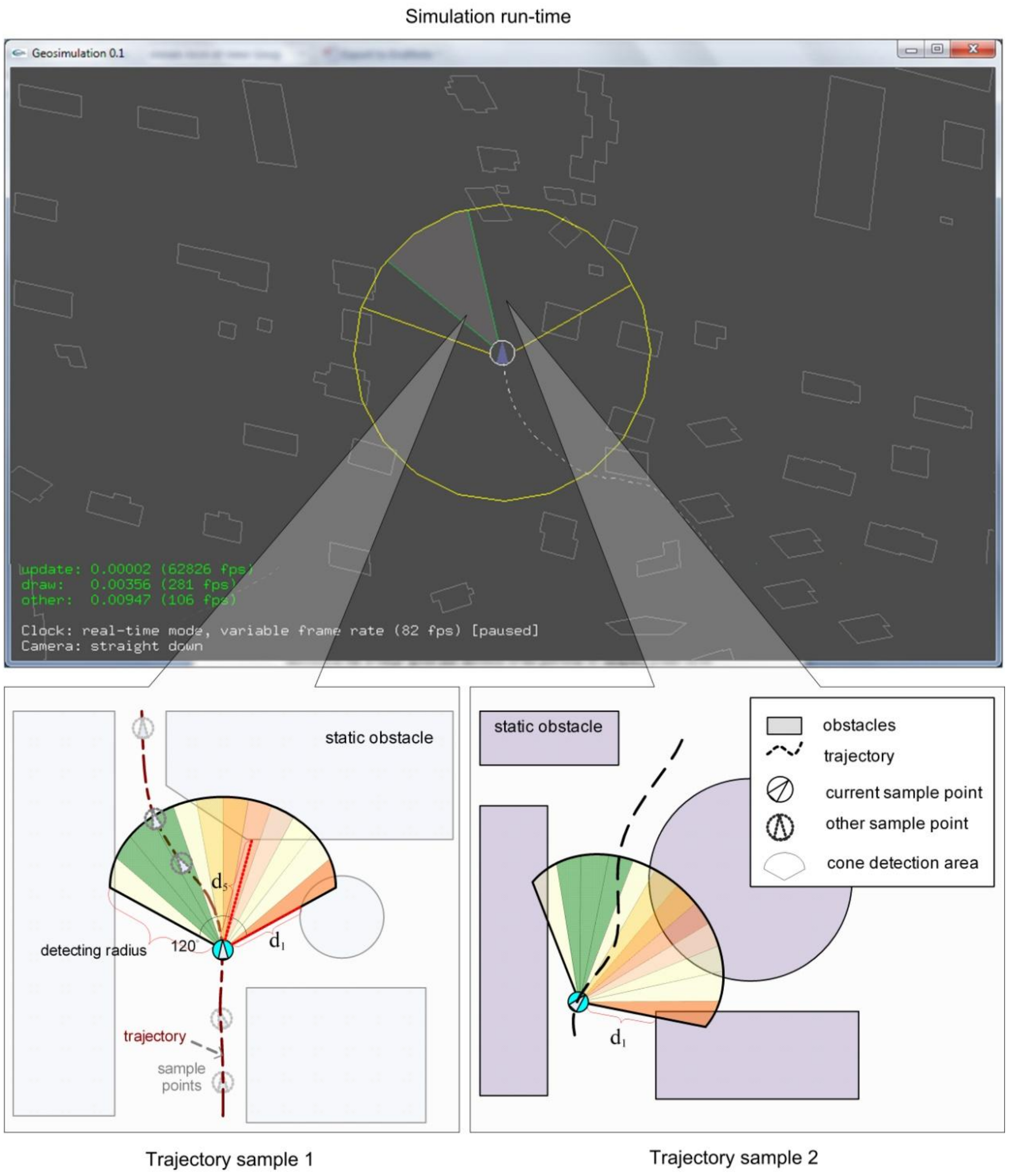


Figure 3: Agent (in-simulation) positioning relative to the local coordinate system. In this local coordinate system, the x-axis denotes the vector of moving direction from a pedestrian's current position to its target and the y-axis represents its direction as a vector from an agent's current location to its destination, rotated by ninety degrees.



```
update: 0.00002 (62826 fps)
draw:   0.00356 (281 fps)
other:  0.00947 (106 fps)

Clock: real-time mode, variable frame rate (82 fps) [paused]
Camera: straight down
```

Figure 4: The environmental information of a sample point. Two candidate trajectory samples are shown at the bottom of the illustration; both may be selected by the model; although their *global* geographies differ, the *local* geographies around the agent are equivalent. Moreover, the agent's trajectories are equivalent (moving toward free space). Both samples can therefore be returned to a querying agent as training data for its movement model. (By comparison, a global model would only be able to return one sample, marrying an agent to that particular scenario.)

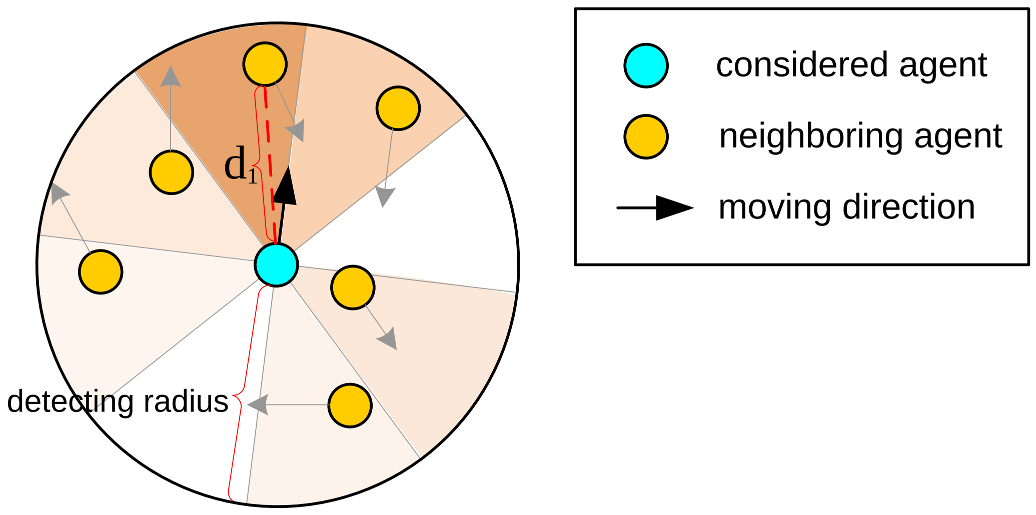


Figure 5: Neighboring humans are polled using a circular awareness zone.

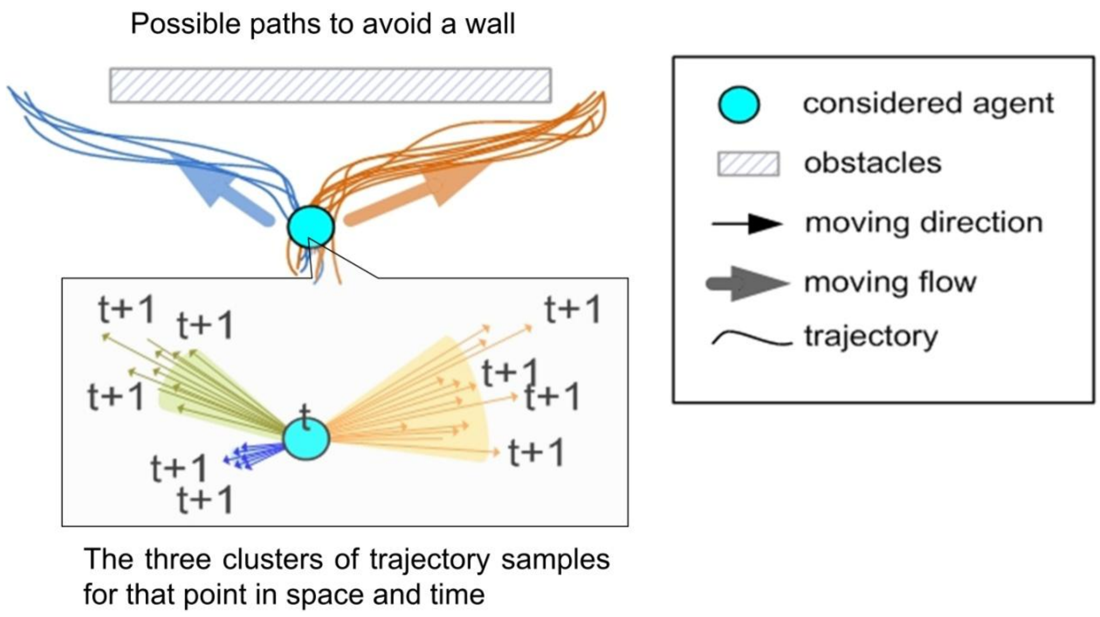

Possible paths to avoid a wall
considered agent
obstacles
moving direction
moving flow
trajectory
t+1
t
The three clusters of trajectory samples for that point in space and time

Figure 6: Choosing clusters from researched samples. Many potential paths can be taken to avoid a wall. Our scheme can sweep a full sample-space of trajectory samples (from real-world choices to avoid the wall) to calculate the most likely future movement.

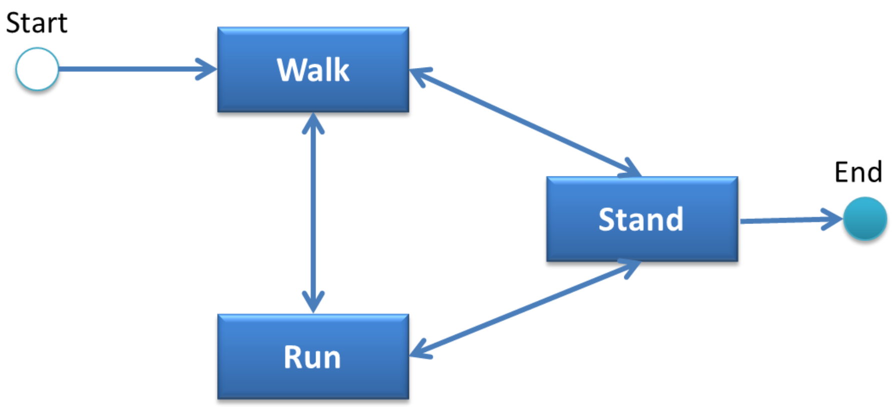


Figure 7: A sample of Finite State Machine as a *high-level control model*.

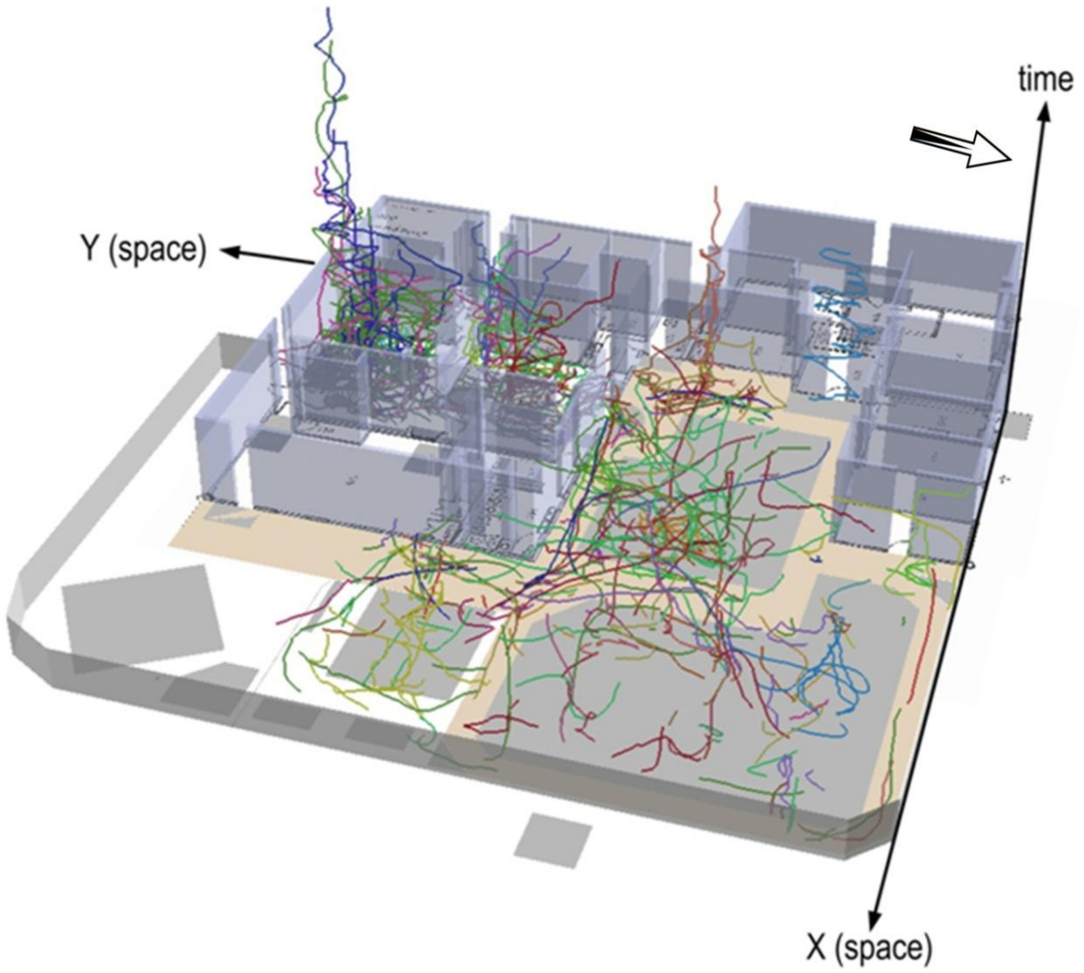


Figure 8 (a)

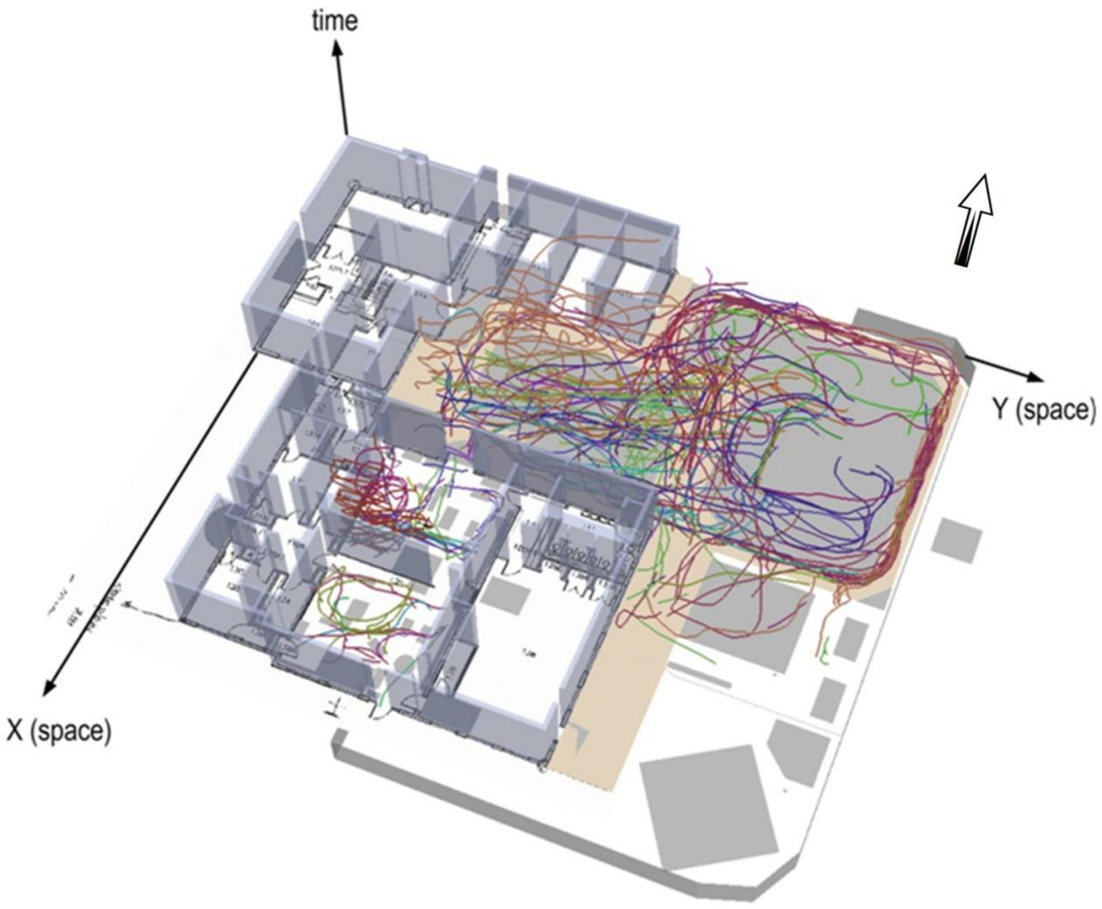


Figure 8 (b)

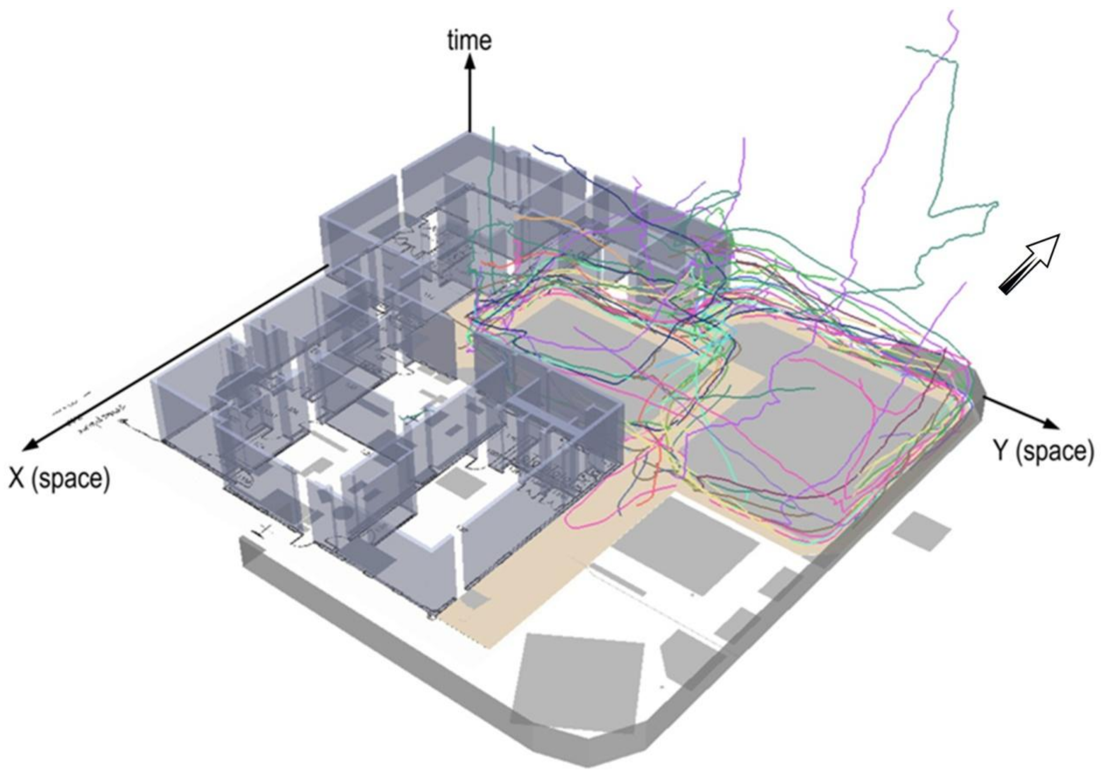
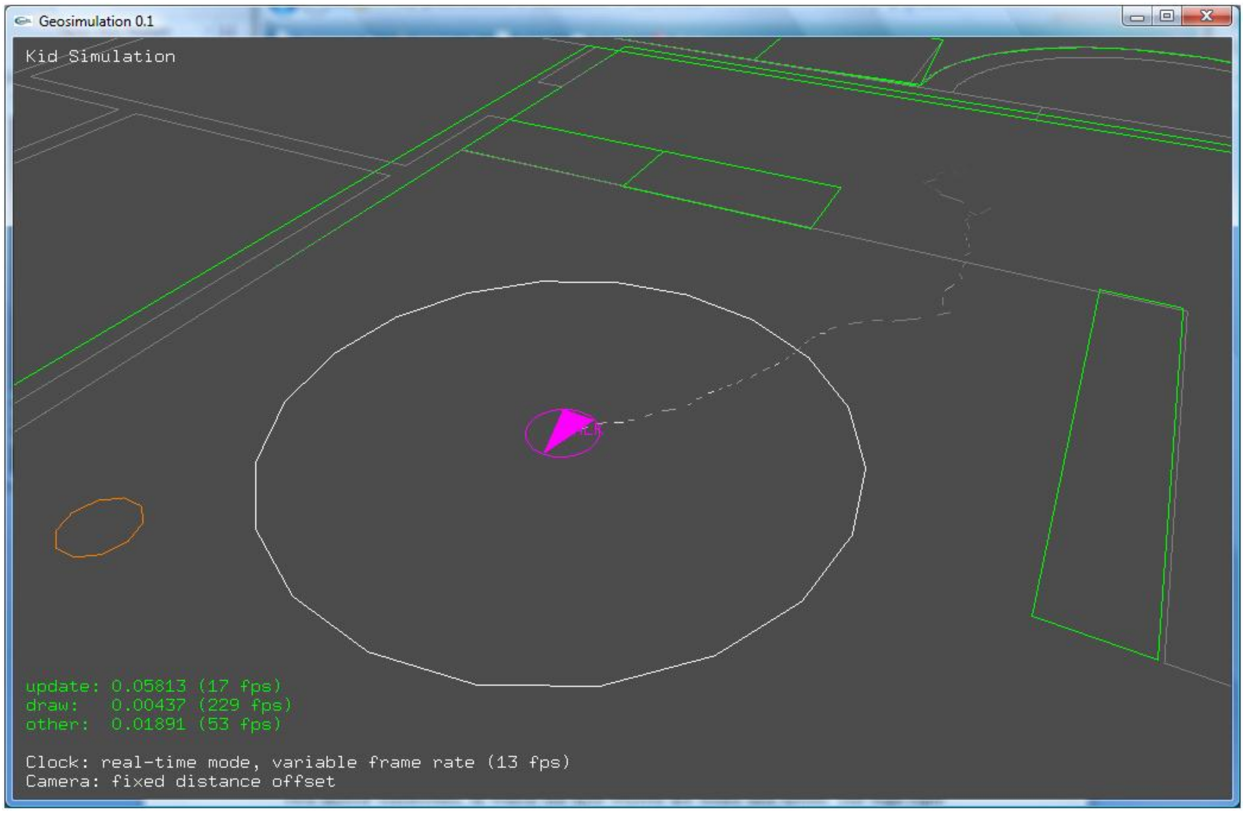


Figure 8 (c)

Figure 8. Sample trajectories for young children in a preschool. (a) Walking; (b) running; (c) riding a tricycle. (Scale bars are inappropriate for 3D graphics; the area represented by the graphics corresponds to 800 square feet on the ground.)



```
update: 0.05813 (17 fps)
draw:   0.00437 (229 fps)
other:  0.01891 (53 fps)

Clock: real-time mode, variable frame rate (13 fps)
Camera: fixed distance offset
```

Figure 9. The children's movement model in run-time. An agent (pink) tries to move between an arbitrary origin and destination (the smaller red circle on the left-hand-side of the illustration, while avoiding collisions with furniture and walls (green) in a simulated classroom. The agent uses a detection filter (white) to poll trajectory samples. Movement is driven solely by our learning scheme and trajectory samples. The run-time environment is rendered using Craig Reynolds's OpenGL libraries (Reynolds 1999) (just the graphic libraries are used, not Reynold's steering behaviors).

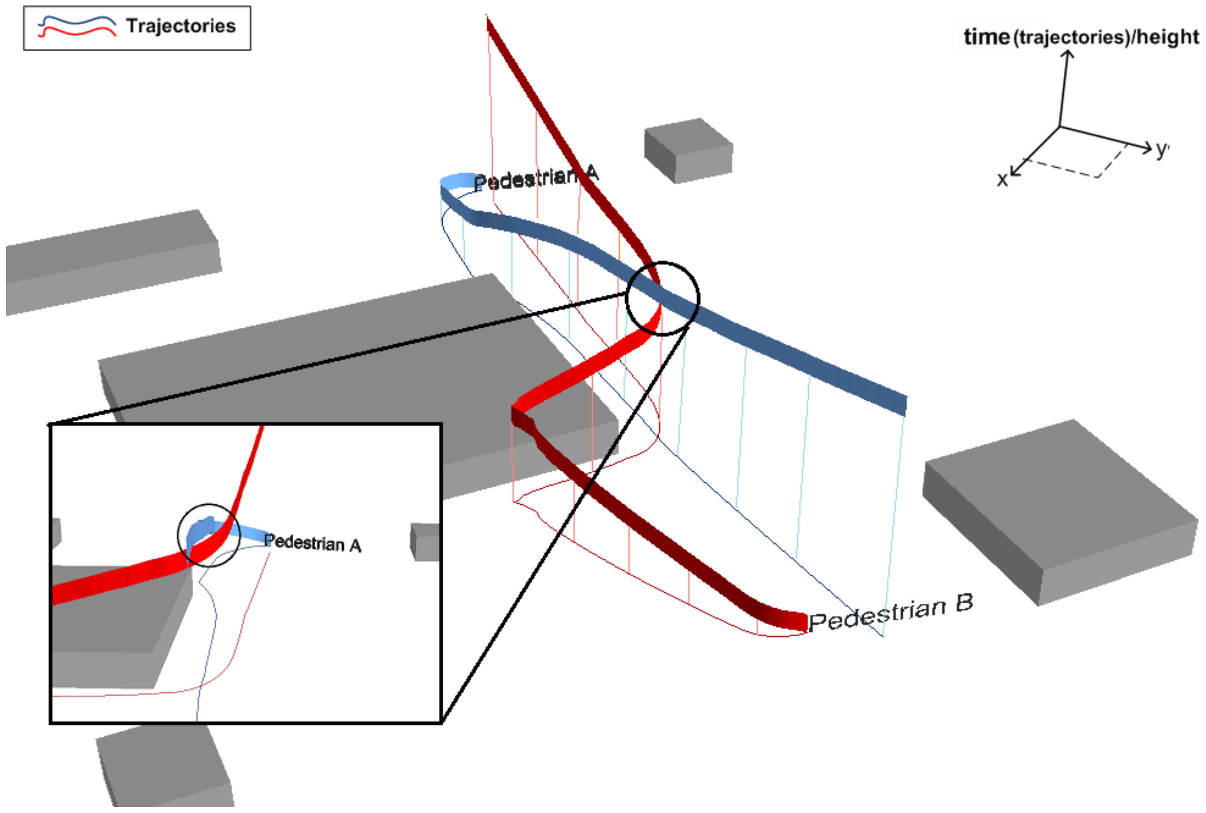


Figure 10. Two space-time paths from the urban pedestrian simulation are illustrated. The space-time paths of pedestrians A and B do not cross, i.e., they avoid colliding. Both agents also avoid built infrastructure in their environment while moving to their destination locations. In the inset, pedestrian B applied steering to avoid bumping into pedestrian A, as evident in the intersecting area of their 3D space-time paths (two dimensions of space and one of time). (The space is simulated as so a scale bar is somewhat irrelevant.)

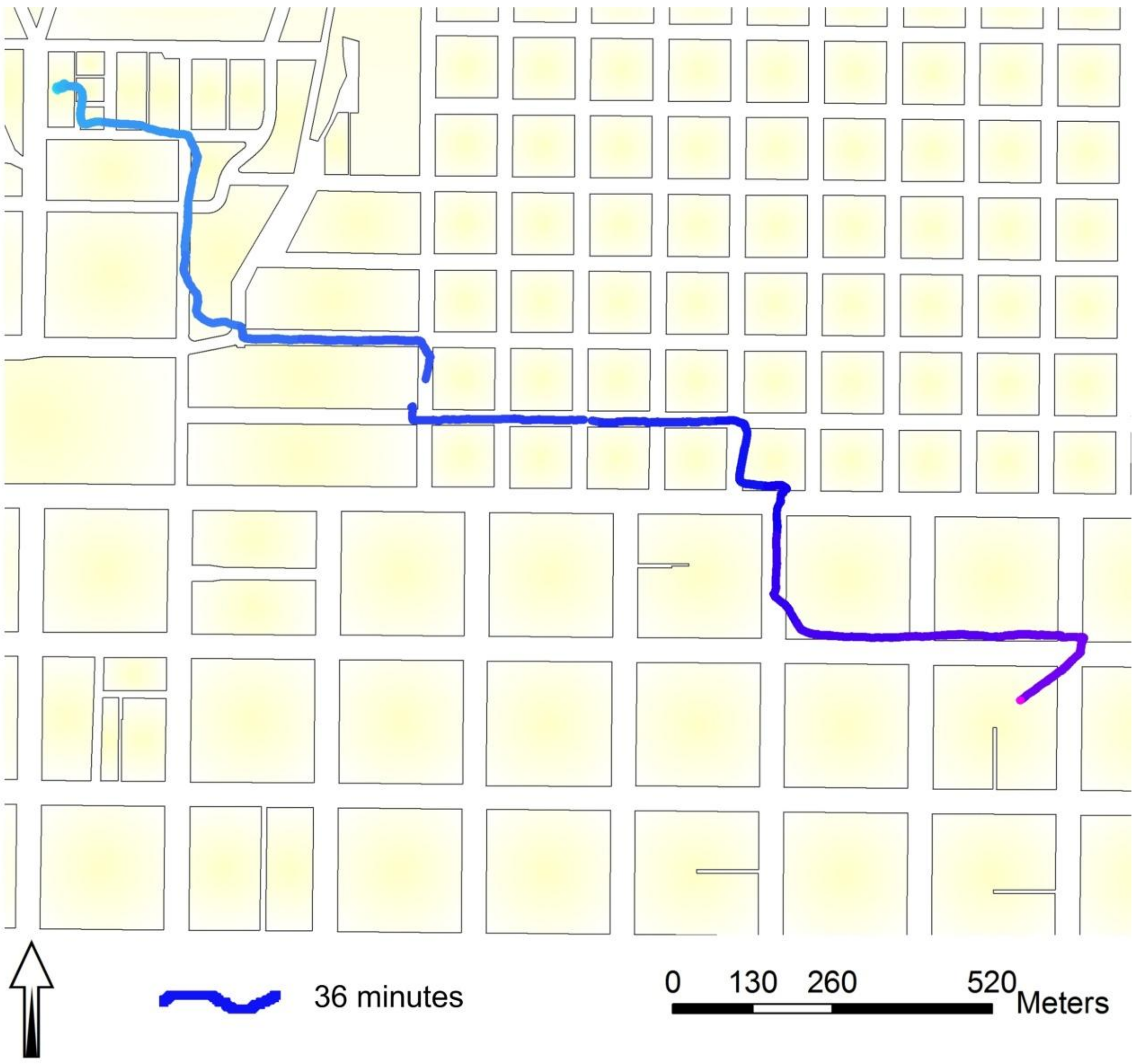


Figure 11. One real-world walking path in downtown Salt Lake City, UT. The path is relatively less sinuous due to low ambient pedestrian traffic. The layout of streets is also more regular than the Yokohama example in Figure 12.

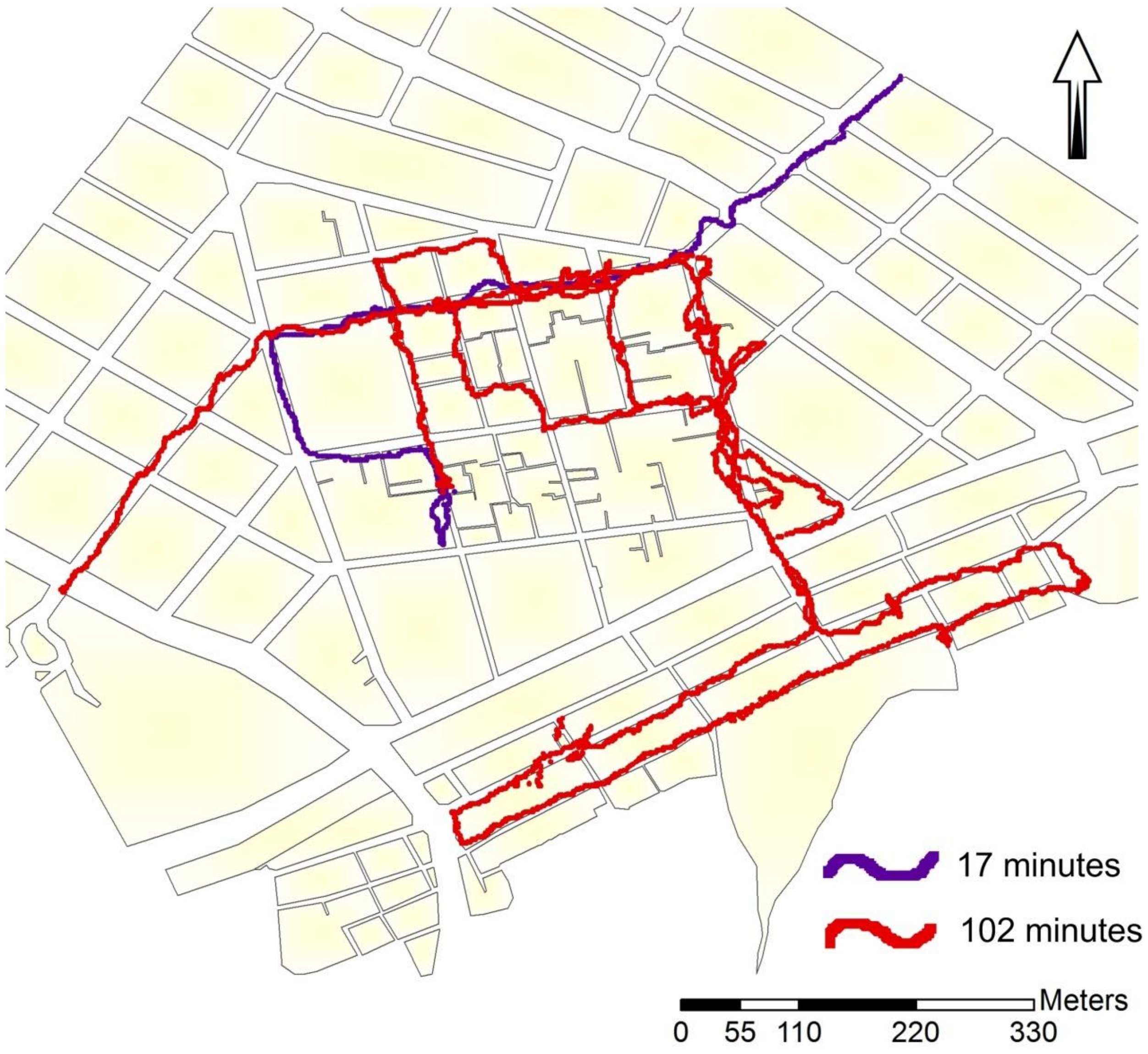


Figure 12. Two real-world walking paths in the Chinatown district of Yokohama. The paths are relatively sinuous because of the dense traffic of pedestrians on the streets. The layout of streets is also more irregular than the Salt Lake City example in Figure 11.

Tables

Table 1: Experimental data. The weight values of different features are empirically tuned to yield realistic behavior.

Learning Models	**Number of samples {M,E,N}**	**Parameters of detection area**	**Weight of motion variables**	**Weight of environment variables**	**Weight of neighborhood variables**
Children's movement (walking)	80,000	r_0=0.5 meters, β=0	1	100	Not applicable
Children's movement (ride bike)	70,000	r_0=1 meters, β=0	1	1,000	Not applicable
Reynolds' capture-the-flag	57,000	r_0=10 meters, β=0.1	1	1,000	Not applicable
Reynolds' collision avoidance	11,000	r_0=8 meters, β=0.1	1	Not applicable	1,000
Urban pedestrians	Combine Reynolds' capture-the-flag and collision avoidance parameters				

Table 2. Movement metrics for sampled and machine-learned paths.

Scenario	$\boldsymbol{D}$	$\boldsymbol{\bar{D}}$	**Scale (min)**	**Scale (max)**	**Length (units)**	**Moves**	$\overline{\cos}\,\boldsymbol{\theta}$	**SD**	$\boldsymbol{corr\ \theta}$	**SE**	**P value**
Sampled real–world movement											
Kids biking (path 1)	1.0604	1.0274	0.07	11.4	30.845	386	0.559	0.489	–0.3923	0.051	0
Kids biking (path 3)	1.0394	1.0169	0.07	20.7	49.229	601	0.77	0.367	–0.6286	0.041	0
Kids biking (path 6)	1.0438	1.0255	0.07	34.8	118.17	1476	0.735	0.398	–0.5199	0.026	0
Kids biking (path 9)	1.0422	1.0178	0.07	24.1	57.35	694	0.787	0.335	–0.5781	0.038	0
Kids biking (path 10)	1.0459	1.0230	0.07	19.8	49.97	591	0.645	0.445	–0.5540	0.041	0
Kids walking (path 2)	1.2024	1.1617	0.08	23.6	144.59	1815	0.636	0.445	–0.4672	0.023	0
Kids walking (path 4)	1.1070	1.0685	0.07	21.6	119.25	1477	0.639	0.425	–0.4517	0.026	0
Kids walking (path 5)	1.0645	1.0366	0.07	24.3	65.91	764	0.823	0.325	–0.5177	0.036	0
Kids walking (path 7)	1.0415	1.0194	0.07	22.7	54.11	612	0.767	0.349	–0.6359	0.041	0
Kids walking (path 8)	1.0499	1.0237	0.07	22.5	63.21	780	0.752	0.381	–0.4662	0.036	0
(Reynolds steering)	1.0092	1.003	0	31.8	63.7	525	1	0	0.6413	0.044	0
Urban pedestrian (American downtown)	1.0356	1.0295	0.01	951.2	2802.1	35062	0.923	0.313	–0.4769	0.005	0
Urban pedestrian (Japanese downtown 1)	1.2084	1.184	0.91	394	6650	6116	0.361	0.684	–0.1536	0.013	0
Urban pedestrian (Japanese downtown 2)	1.1565	1.1037	0.91	287.1	1215.6	1047	0.463	0.599	–0.2499	0.031	0
Machine-learned movement											
Machine-learned walking	1.0331	1.0164	0.01	8.01	18.69	167	0.905	0.23	–0.0167	0.078	0.831
Machine-learned tricycling	1.0036	1.0095	0.02	22.31	71.83	269	0.981	0.043	0.0112	0.061	0.855
Machine-learned capture the flag	1.0458	1.0056	0.01	18.75	42.76	286	1	0	3.6671	0.059	0
Machine-learned urban pedestrian	1.0527	1.023	1	100	81.84	227	0.993	0.075	0.2934	0.067	0

SD: standard deviation of the metric to the left; SE: standard error of the metric to the left. Reynolds steering data is simulated (not learned).

Table 3. Entropy metrics for real-world paths.

Scenario →	Kids bike 1	Kids bike 3	Kids bike 6	Kids bike 9	Kids bike 10	Kids walk 2	Kids walk 4	Kids walk 5	Kids walk 7	Kids walk 8	Salt Lake City	Yokohama 1	Yokohama 2
Entropy metric ↓													
AppEn velocity	0.47	0.38	0.82	0.37	0.63	0.98	0.9	0.5	0.27	0.47	0.18	0.87	1.02
AppEn acceleration	0.6	0.34	0.89	0.37	0.62	1.05	0.85	0.56	0.24	0.42	0.14	0.58	0.6
AppEn direction	0.44	0.3	0.9	0.31	0.6	1.12	0.9	0.72	0.41	0.53	0.18	0.17	0.16
Mean velocity	2.77	6.21	4.67	7	4.43	3.58	4.32	5.89	7.4	4.93	0.78	1.09	1.16
Velocity standard deviation	1.92	2.58	2.86	2.3	3.37	2.43	2.5	3.39	3.62	2.18	1.28	1.21	1.07
Velocity skew	1.14	0.54	1.05	0.68	1.2	1.49	1.13	0.63	0.44	0.36	17.4	3.29	1.03
Mean acceleration	0.15	–0.66	–0.66	0.38	–0.67	–0.1	–0.24	–1.35	–0.7	–0.09	0	0	0
Acceleration standard deviation	14.43	21.31	19.63	22	24.3	15.85	17.47	28.9	27.29	19.68	2.08	2.3	2.19
Acceleration skew	–0.23	0.08	–0.31	–0.4	–0.89	0.15	–0.33	0.2	–0.18	–0.14	–2.5	0.83	–0.06
Mean direction	–0.01	–0.01	–0.1	–0.1	0	0	0.03	–0.04	–0.01	–0.01	0	–0.02	–0.01
Direction standard deviation	0.35	0.18	0.29	0.16	0.25	0.31	0.28	0.33	0.21	0.23	0.24	0.52	0.5
Direction skew	0.29	–0.2	0.11	–0.44	0.52	2.54	2.18	–3.8	–0.47	0.6	0.42	–1.17	–0.71
Trajectory length	26.14	45.06	106.42	52.8	44.16	125.9	104.31	61.23	49.88	57.08	2798	6650	1215
Trajectory duration	9.44	7.25	22.78	7.55	9.97	35.31	24.14	10.34	6.7	11.56	3575	6122	1047

Trajectory length is in meters; trajectory duration is in seconds.

Table 4. Entropy metrics for machine-learned paths.

Scenario →	ML walking	ML cycling	ML capture the flag	ML urban pedestrian	Reynolds steering
Entropy metric ↓					
AppEn velocity	0.65	0.97	0	0.57	0.03
AppEn acceleration	0.82	1.13	0	0.52	0.25
AppEn direction	0.71	1.03	0.03	0.16	0
Mean velocity	2.24	2.7	2.4	6.75	3
Velocity standard deviation	1.3	1.26	0.9	1.5	0
Velocity skew	1.62	0.83	–2.3	–0.02	–13.26
Mean acceleration	0.34	0.11	0.08	0.29	0
Acceleration standard deviation	24.9	18	1.76	24	0.06
Acceleration skew	1.21	–0.05	17.17	0.08	–9.14
Mean direction	–0.04	0	7.6	0.01	–0.01
Direction standard deviation	0.5	0.23	0.01	0.03	0.01
Direction skew	–2	0.17	–0.22	2.07	–1.4
Trajectory length	18.69	72.34	63.7	76.5	42.76
Trajectory duration	8.35	26.82	26.22	11.33	14.25

ML: machine-learned. Reynolds steering is simulated (not learned).