Volume xx (200y), Number z, pp. 1–14

The PAG Crowd: A Graph Based Approach for Efficient Data-Driven Crowd Simulation

P. Charalambous and Y. Chrysanthou

Department of Computer Science, University of Cyprus, Cyprus
{totis,yiorgos}@cs.ucy.ac.cy

Abstract

We present a data-driven method for the real-time synthesis of believable steering behaviors for virtual crowds. The proposed method interlinks the input examples into a structure we call the Perception–Action Graph (PAG) which can be used at run-time to efficiently synthesize believable virtual crowds. A virtual character's state is encoded using a temporal representation, the Temporal Perception Pattern (TPP). The graph nodes store groups of similar TPPs whereas edges connecting the nodes store actions (trajectories) that were partially responsible for the transformation between the TPPs. The proposed method is being tested on various scenarios using different input data and compared against a nearest neighbors approach which is commonly employed in other data-driven crowd simulation systems. The results show up to an order of magnitude speed-up with similar or better simulation quality.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation—I.2.11 [Distributed Artificial Intelligence]: Multiagent systems—

1. Introduction

Virtual crowds are important in a variety of applications such as computer games, movies, training simulations and safety modeling. Increasing processing power due to multicore architectures, improved clock speeds and highly programmable Graphics Processing Units (GPUs), enable designers and programmers to add multitudes of virtual characters in real-time applications. As the real-time rendering of the characters is becoming more and more realistic, there is a considerable gap between the rendering appearance and the simulated behavior of crowds. People are accustomed to real people in their everyday lives and therefore can easily differentiate between a virtual human and a real one from their behavior patterns, even though virtual humans can be modeled and rendered realistically.

Numerous crowd simulation algorithms have been proposed over the last few years. These algorithms can be divided into two general categories: macroscopic, that consider the simulation of the crowd as a whole and are mostly interested in a global view of the crowd, and microscopic in which global crowd phenomena emerge by simulating each individual virtual person's behavior separately. One of the most common forms of the microscopic approach models the virtual humans as agents and their behavior is defined by a set of rules. Such an agent-based system can retain character variability as seen in a real crowd while at the same time provide the ability to control the agents individually through personalized higher-level decision making.

Specifying the rules that govern agent behaviors is a very laborious and demanding task. Data-driven methods avoid this overhead by following rules indirectly embedded in the real crowd data. Such data is usually extracted from videos in which the people's trajectories are tracked and processed. Several crowd datasets already exist freely available on the Internet, while with the rapid progress in Computer Vision they become increasingly easier to generate. Despite the attractiveness of data-driven techniques, they do carry a much higher computational cost, at simulation time, compared to traditional rule-based ones. The process of querying a typically high dimensional example database and forming the actions for all the agents at every simulation step is costly even if efficient acceleration structures are employed. Additionally, most of these algorithms do not try to enforce temporal consistency between steps.

Inspired by previous work on Motion Graphs [KGP02, PP10], we propose the *Perception-Action Graph (PAG)* as a method for accelerating and improving the quality of data-driven crowds. Preprocessing the input data into forming the PAG is key since it offers significant advantages at simulation time over traditional techniques:

1. Simulation **run-time performance** is improved significantly since the PAG groups together similar states and interconnects them with actions, reducing the need for continuous database searches.
2. Simulation has improved **adaptation** to the source data since action selection is directly influenced by previous states and action; not just the current state as in most traditional data-driven approaches.
3. Since the PAG is constructed at preprocessing, more accurate and expensive comparison methods can be employed for finding and grouping similar states or even for learning new actions (such as the regression-based action selection mechanism by Lee et al. [LCHL07]) instead of doing so at run-time.

The proposed method has been tested on a number of scenarios with different input data and the results show that when compared to a method similar to the one employed by Lerner et al. [LCL07], performance increases by up to an order of magnitude and quality is at least similar.

2. Previous Work

Crowd simulation is concerned with the simulation of groups of virtual characters which typically have common goals and characteristics such as people in a parade, flocks of birds and fans in a stadium. As a field, crowd simulation has been extensively studied and applied by various scientific areas such as psychology, computer graphics, safety engineering, etc. Good overviews of the area can be found in the books by Thalmann and Musse [TM07] and Pelechano et al. [PAB08].

Over the years, a large number of different methods have been proposed in the literature taking widely different approaches. Some of the early works such as the seminal work by Reynolds [Rey87] and many that followed thereafter [Rey99, LD04, ST05] employ rule-based approaches; individual characters take into account a local view of their state and act upon it. Others borrow ideas from fluid mechanics [Hug03, TCP06], while others make use of particles and social forces [HM95, HLTC03, POSB05]. Although some of these methods are quite scalable and can capture the aggregate behavior of a crowd fairly well [TCP06, NGCL09], they cannot fully capture the range or the subtleties of individual behaviors. Complex behavior patterns can be implemented in some rule based systems [Mas07] by defining many finely tuned situation-specific rules which are in most scenarios quite difficult and laborious to define by a non-expert.

Data-driven techniques have been extensively used in many areas of computer graphics. For example, many recent texture synthesis techniques are able to synthesize large textures from small examples [KSE*03] and fill in holes in images [DCOY03]. The image analogies approach uses examples to learn about and reproduce relationships between image pairs [HJO*01]. Other applications include surface completion [SACO04], image colorization [ICOL05] and image segmentation [SCCOL06]. Zhang et al. [ZCM11] employ an idea similar to motion graphs [KGP02] for fire simulation.

Recently, data-driven crowd simulation methods have emerged as an attractive alternative to manually defining the crowd simulation model. The promise in these approaches is that agents will "learn" how to behave from real-world examples, keeping the natural crowd ambiance with a wide range of complex individual behaviors without the effort of defining an explicit behavioral model. One of the earliest data-driven techniques for groups of characters employed a motion graph approach for synthesizing group behavior [LCF05]. In order to be able to build a tractable motion graph, this method makes the assumption that the input follows a well defined behavior model, such as a flocking system with a restricted configuration space; i.e., the group has a constant number of agents and is simulated as an entity. Graph based simulation was also used by Kwon et al. [KLLT08] for guiding a single group of agents navigating together. The great variation in the movements of a general human pedestrian crowd would render these methods impractical.

In recent works [LCHL07, LCL07] trajectories learned from videos of crowds are stored in a database alongside some representation of the stimuli that affected them. During simulation, agents match their stimuli to the ones stored in the database and navigate accordingly. Following from these, Lerner et al. [LFCCO09] used a database approach to add secondary actions to simulated characters such as talking or looking at their watches. Ju et al. [JCP*10] take input data that represent different styles of crowds and blend them to generate new crowd animations. Metoyer and Hodgins [MH03] allow the user to define specific examples of behaviors, while Musse et al. [MJBJ06] extract paths from a video for a specific environment.

As an alternative to employing databases of example situations, some techniques use observations of real people to extract simulation parameters. Several works [PPD07, POO*09] estimate collision avoidance and anticipation parameters by examining motion capture data in a controlled environment and propose prediction based approaches for crowd steering. Moussaïd et al. [MPG*10] used data from videos of real crowds to modify Helbing's social forces model [HM95] to handle group formations in a more realistic way. In the work of Courty and Corpetti [CC07] the crowd is seen as a continuous flow and the captured data are used to define the guiding vector field. Looking a bit further away, biology researchers [HCH10] proposed using input from stereoscopic videos of Starling birds to estimate a statistical model of their massive and complex flocking behavior. In all of the techniques described here, the examples are used to refine an underlying behavior model therefore they are still bound by the limitations of the model.

Very relevant to our work are the data-driven methods used very successfully in animation and motion synthesis for single or two interacting characters [KGP02, HPP05, RSH*05,PP10,KCPS08,SKY08]. In these techniques, a motion graph is constructed that interlinks transitions between postures of one or two characters whereas in the proposed method connections are made between perceptual information of multiple characters.

In typical data-driven crowd steering methods such as the ones by Lerner et al. [LCL07] and Lee et al. [LCHL07], a (state, action) database is queried multiple times per simulation step, potentially once for each character, in order to retrieve the best matching examples and then the best possible action is found. These queries are usually the bottleneck due to both the high dimensionality of the state representation and the number of results that are returned. High dimensionality searching can typically be addressed using approaches such as Principal Component Analysis (PCA) that infer some penalty on accuracy. Even though search performance is improved, selecting the best action is still costly since all of the returned results need to be processed. In addition, such approaches assume that crowd steering is more or less a Markov process; decisions are based on the current state of agents or at most the previous step's decision and not previously taken actions and states. In order to improve both run-time speed and quality for data-driven crowd simulations, we propose a method that groups similar states and interconnects them in a graph like structure, the *Perception-Action Graph (PAG)* that is employed at simulation time. The proposed approach differs from the approach followed by Ju et al. [JCP*10], where given a set of crowd data (simulated or real), new and potentially much larger crowds are generated by interpolating between them.

3. Overview

As in previous data-driven methods, a person's behavior is assumed to be affected mainly by external stimuli such as other characters and objects. It is acknowledged that internal stimuli emanating from a person's state of mind, mood, beliefs, higher level goals and so on, are also important but these are hard to observe in the input data and are therefore neglected. More specifically, in this work it is assumed that steering behaviors, such as turning, accelerating or stopping depend on the flow of perceptual information a person observes; similar to the synthetic vision approach followed by Ondřej et al. [OPOD10]. Input data can be obtained either from videos of real life crowds or extracted from crowd simulations. The proposed methodology can be used to control the low level steering behavior of individual characters, who might or might not have a desired short term target destination. If needed, the virtual characters can be further orchestrated by an independent higher level controller/path planner that assigns goals at run-time.

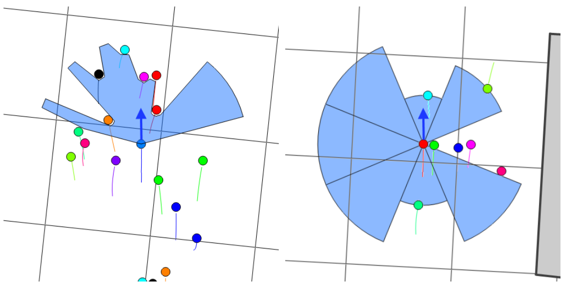

(a) Visibility Based (b) Radial Based

Figure 1: In (a) instantaneous state is represented as the agent's visibility whereas in (b) the agent's surrounding area is divided into radial regions and the distance to the closest neighbor is found for each (similar to [LCHL07]).

The proposed framework operates in two phases; *preprocessing* and *run-time simulation*. At the preprocessing stage, similar situations are identified, clustered and interconnected in a graph structure (the PAG) exploiting possible transitions from one perceptual state to another. Each cluster stores also the different actions taken by people due to those situations. Additionally a database of actions that were taken in interaction free (IFDB) situations is generated. At run-time, simulated agents employ both the PAG and the IFDB to efficiently simulate behavior similar to the source data.

3.1. Data Preprocessing

The trajectories of people in the input data are tracked and sampled at regular intervals (Figure 2). Each sample encodes the instantaneous state of a person relative to its local coordinate system; i.e., with the person centered on the origin facing alongside the positive Z-axis. A set of N consecutive instantaneous samples on the same trajectory (a clip of 0.5–2 seconds) represent a temporal example situation encountered by the person (we call this the *Temporal Perception Pattern* – the TPP) which practically encodes the flow of information he observes (Figure 4).

All these examples are preprocessed and linked together into the PAG data structure. The PAG is a directed graph $G = (V,E)$ in which a node V_i represents a group of similar TPPs and an edge $E_{i,j}$ represents an action $A_{i,j}$ that was partially responsible in transforming the TPP stored in node V_i to the one in node V_j (Figure 5). An action is defined as a trajectory segment followed by a pedestrian in the input.

3.2. Simulation

A simulation scenario consists of the static environment alongside the initial configuration of the characters (posi-

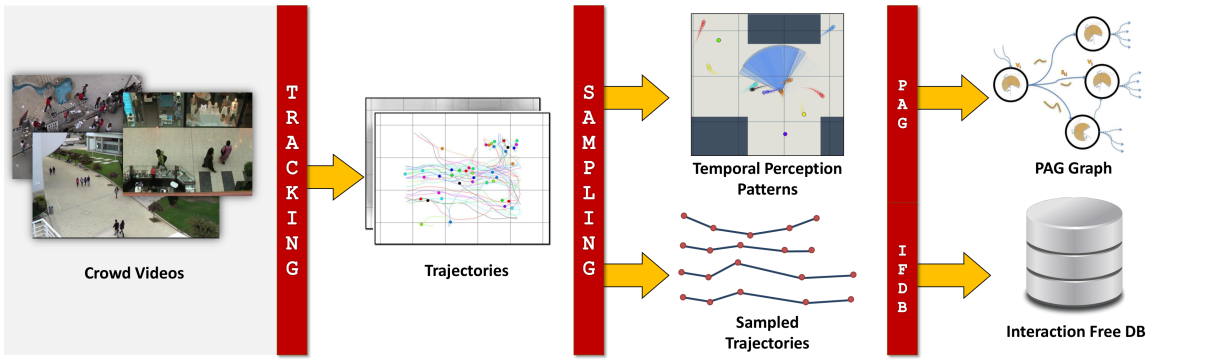


Figure 2: Preprocessing pipeline. People from videos of real life crowds or expensive simulations are tracked, trajectories are extracted and sampled resulting in the PAG generation alongside a database of interaction free trajectories (IFDB).

tions, velocities, goals). Each character is represented as a circular agent. During simulation, agents perceive the environment at regular intervals to generate TPPs similarly to the preprocessing step. If the agents sense no stimuli in their view, they simply move towards their goals by selecting appropriate trajectories from the Interaction Free DB. Whenever interactions are sensed (i.e., some external stimuli enters the sensing area of the agent), agents search the PAG to find the best matching node as a starting point to handle the interactions (Figure 3).

Graph traversal leads to behaviors such as collision avoidance, stopping to talk to each other, following someone, etc. Neighboring agents coordinate to select the best actions at each simulation step by performing constrained walks of the PAG. Best actions are defined as the ones that minimize the error between actual agent states and those stored on the currently traversed nodes of the PAG (Figure 8).

A more detailed description of the preprocessing and simulation phases follows in Sections 4 and 5 respectively.

4. Graph Construction

In this work, people's temporal state (the TPP) is visibility based; i.e., behavior is directly correlated to information from a person's view field, similarly to the approach by Ondřej et al. [OPOD10]. This approach was employed since sensory information from viewing is typically much larger than from the other senses [Nør91] and plays a large role in action selection. It is important to emphasize here that the underlying method can be used with other state representations also as long as the state vector is encoded using a constant number of values (e.g., the approaches in Figure 1).

Visibility Visibility encodes the free space in the field of view (FOV) of a person; i.e., the space that is not occluded by other people or static objects as this is projected on the floor. People are modeled as circles projected on the ground plane. The FOV is aligned to a person's moving direction (Figure 1a) and is sampled at regular angular intervals. For each angular sample the distance to the closest object (static and dynamic) is found using either ray-tracing or a z-buffer style approach. The sampling interval is dependent of the FOV width, the search radius and the size (radius) of the agents; dense sampling gives better approximation of the visibility at the expense of performance and memory stor-

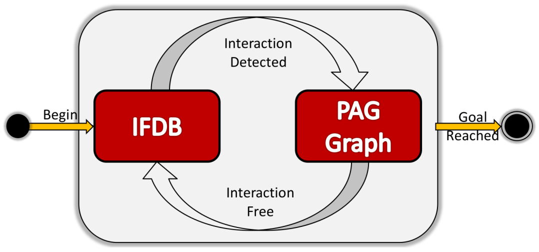


Figure 3: Agents' state diagram. An agent moves towards its goal by using the interaction free trajectory DB. Whenever an interaction is detected; it utilizes the PAG to resolve it.

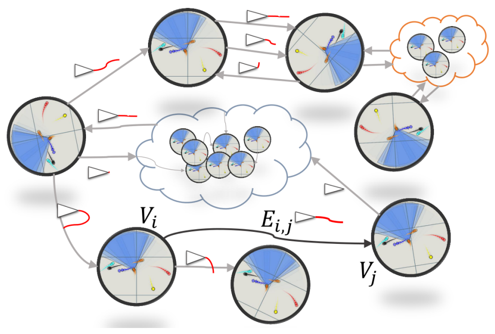


Figure 5: A PAG represents transitions from one TPP to another. An edge $E_{i,j}$ between two nodes V_i and V_j represents an observed transition. This transition was partially the result of a trajectory (red lines) followed by a person with a TPP similar to the one stored in V_i.

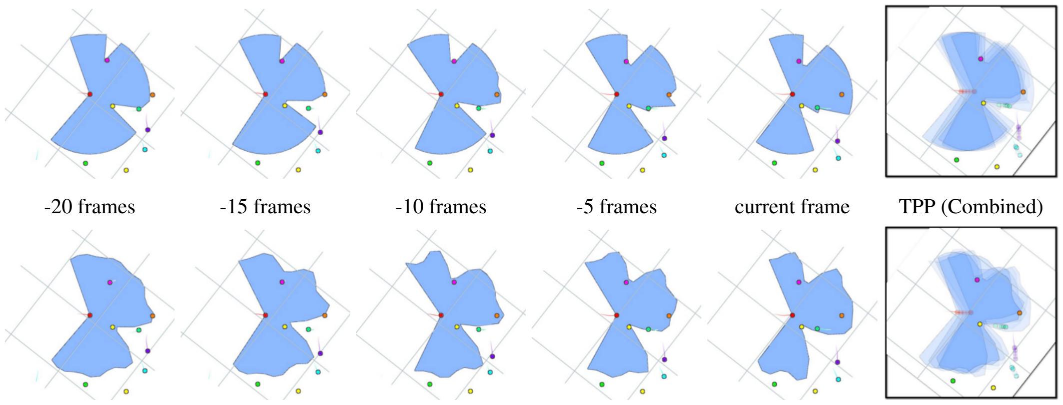


Figure 4: A *Temporal Perception Pattern (TPP)* consists of a set of consecutive in time visibility patterns. (top row) Five equally spaced in time visibility patterns define a TPP. (bottom row) A compressed version of the TPP.

age. For most of the conducted experiments, the FOV ranged between 90° to 240° and was sampled using 20 to 40 rays.

TPP A predefined number of consecutive trajectory samples (i.e., a clip of instantaneous states) are grouped together and form a *Temporal Perception Pattern* (Figure 4). By using *temporal* information, the relative positions and actions of people are encoded indirectly; i.e., without any explicit definition of each individual's velocity. TPPs without any stimuli are assumed to be interaction free and are added in a database of interaction free cases (IFDB) (Figure 2). The remaining TPPs are inserted into a Perception Action Database (PADB) of $(state, action)$ examples where states are represented by TPPs and actions are trajectory segments. Each state is of relatively large dimensionality; if for example a trajectory is sampled 5 times per second and for each sample the visibility is represented with 20 values, $1s$ of state is encoded using 100 values.

The PADB could then be queried by each agent during simulation to extract the best matching examples using a nearest neighbor algorithm and then apply the retrieved actions, either directly or after further processing. This approach is the most commonly used in the data-driven literature for crowds [LCL07, LCHL07, TLG11]. Instead of following this expensive practice, a different approach is employed here: similar TPPs are identified, grouped together and interconnected indicating transitions between them to form the PAG, a directed graph which is used to speed up the simulation in a reliable manner.

In Sections 4.1 and 4.2 a description of the similarity metric and the graph generation algorithms are presented.

4.1. TPP Similarity

TPPs are grouped together to form the PAG using an approach conceptually similar to that employed in single character animation. In Kovar et al. [KGP02], good transition points between poses are found using a metric that takes into account both joint positions and orientations over a time window. The distance metric in this work measures similarities between TPPs: for each pair of agents (A, B) and each pair of TPPs $(P_{A,k}, P_{B,m})$ at frames k and m respectively, their distance is found using a correlation metric and a normalized distance matrix is calculated for all sampled frames (Figure 6) (with 1 indicating maximum and 0 minimum similarity respectively). For an input dataset of N pedestrians, $N^2/2$ distance matrices are generated due to symmetric properties of the data.

Unlike pose comparisons where the number of joints is constant, the number of stimuli in an agent's TPP might vary from 0 to any number. To measure similarity between TPPs, a metric is required that has a constant number of features and in addition compresses well and is insensitive to small changes, especially in the distant stimuli. The proposed metric is based on smoothed out versions of the TPPs using the lower frequency coefficients of its type-II DCT (Discrete Cosine Transform) [ANR74]: each visibility pattern of the TPP is considered as a signal $x_j(\theta)$, where $x_j : [-\theta/2, \theta/2] \longmapsto [0, r]$, θ is the field of view of the agent and r is the maximum search radius of the agent. By concatenating all m visibility signals x_j of the TPP, we get a signal representation of the whole TPP: $x = (x_1, x_2, ..., x_m)$. This signal is sampled at regular intervals to generate the sampled signal $x' = \{x_i : i \in [0, N-1]\}$. The DCT of this signal is $X = \{X_k : k \in [0, N-1]\}$ where:

$$X_k = \sum_{i=0}^{N-1} x_i \cos\left[\frac{\pi}{N}(i + \frac{1}{2}k)\right] \quad (1)$$

Out of these N coefficients (X_0 to X_{N-1}), a small number M $(M < N)$ of the lower frequencies are enough to approximate the original TPP quite well. For example, the TPPs in Figure 7 can be approximated using just $8 - 12$ DCT coefficients despite the TPP complexity. Additionally, the TPP comparison is more stable since data is de-noised at the

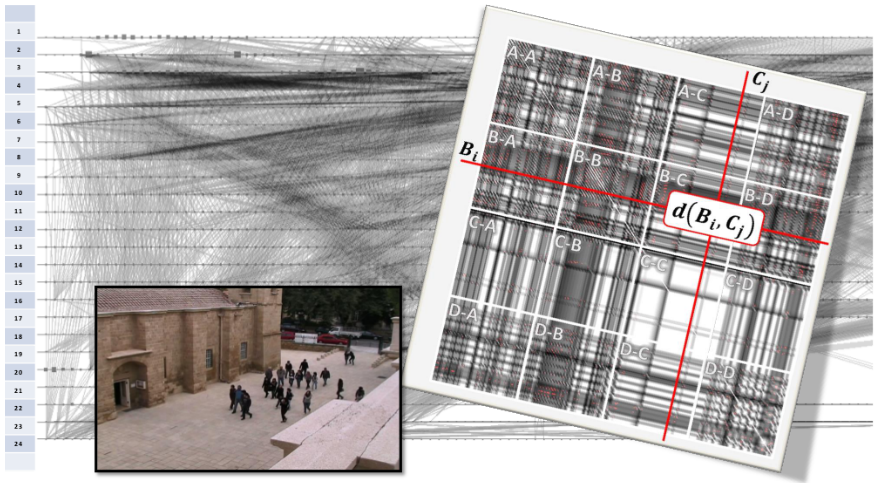


Figure 6: Similarity map for a video of a "flock" of 24 people in a tourist site (left image). Horizontal lines indicate the trajectory of a tracked pedestrian over time whereas diagonal lines interconnect points where pedestrians had similar TPPs as these were found by distance matrices (right image).

boundaries of the visibility. In effect, this approach smooths out the distance matrix. To compare two signals x and y, the Root Mean Squared Error (RMSE) between their lowest $M \in [8,12]$ DCT frequency coefficients is found and normalized to $[0,1]$. The RMSE is found at the frequency domain to reduce calculation cost, since the energy of a signal over its entire domain is equal to the energy of its Fourier based transform over all the frequencies [PdC06]. The correlation of the two signals is calculated using Equation 2:

$$CORR(x,y) = 1 - \sqrt{\frac{1}{N}\sum_{i=0}^{M-1}(X_i - Y_i)^2} \tag{2}$$

X and Y are the DCT transformed signals of x and y respectively. One row of the distance matrix indicates the similarity of a specific TPP to all other TPPs in the input data and the local maxima of each row indicate the best matches of that TPP. This approach differs slightly from the one followed by Kovar et al. [KGP02], where the best transition points are found in a local neighborhood since it generates relatively well connected graphs. Out of these maxima, the best over a threshold are selected (typically > 0.9); increasing this threshold value results in increased quality but with fewer interconnections and therefore less choices during simulation (Section 6). Table 1 lists thresholds for the experiments presented in this work.

4.2. Temporal Perception Patterns Interconnection

Once the best matches are found, a directed graph (the PAG) connecting transitions between TPPs is generated (Figure 5). In Figure 6, a semantically similar graph to the PAG displaying the connections between similar TPPs can be seen. Each node V_i of the graph groups together similar TPPs and an edge $E_{i,j}$ between nodes V_i and V_j is created when a TPP in node V_j was a followup of a TPP belonging in V_i. These TPPs are temporally overlapped; i.e., the starting section of the TPP in V_j is similar to the ending section of V_i. The edges of the PAG store trajectory segments that the pedestrian in the source data followed. A node has as many edges as the number of TPPs clustered there. Potentially, node V_i can have multiple edges pointing to node V_j but each edge can store different actions (trajectories). After the nodes are interconnected, the PAG is post-processed to remove dead-ends so that the graph can be used at simulation time without any special handling.

5. Simulation

The PAG graph can be viewed as a temporal state-action database. Obviously, if simulated agents follow blindly random edges of the graph, the emerged steering behavior will not be realistic since agents will not be taking into account other agents and their actions. Additionally a person's perception changes not only because of his own actions but also due to other people's decisions; a fact that should be taken into account during simulation. TPPs observed during simulation will most likely differ from the ones agents observe on the currently visited nodes of the PAG and therefore agents should coordinate so that dynamically generated TPPs match the ones stored on the graph nodes.

A constrained walk on the PAG graph is proposed: whenever an agent senses a possible interaction with other agents or obstacles it selects the node on the graph that mostly resembles its current state as a starting position for interaction handling. For as long as there are interactions, the agent traverses the graph coordinating with neighboring agents and

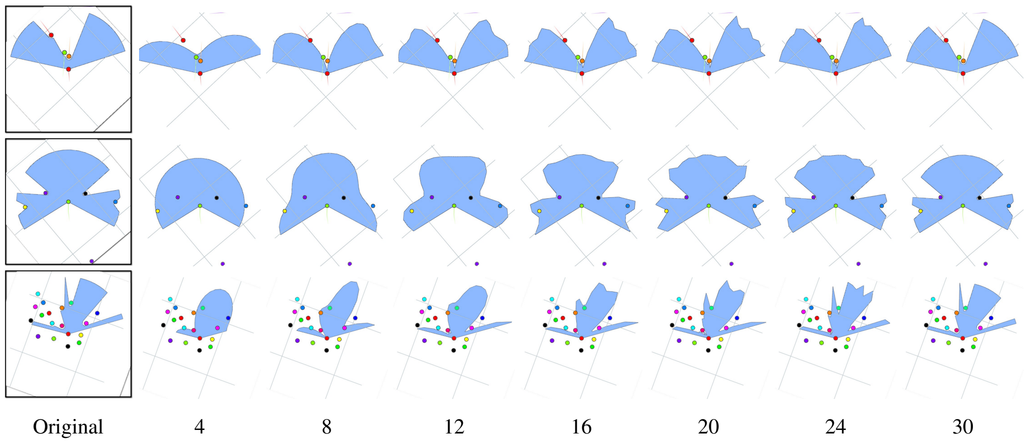


Figure 7: The effect of the number of DCT coefficients (bottom row) on visibility based TPP compression for various cases. Here only one visibility pattern of the TPP is shown. By using 8-12 coefficients, the original pattern is estimated without much loss in accuracy.

Algorithm 1 Crowd Simulation - Update Stage

```
Input:
$A$: Set of Agents $A = \{A_i : i \in [1,N]\}$
$k$: Number of nearest examples
Output
$A$: Set of Agents $A = \{A_i : i \in [1,N]\}$ with updated positions
Update($A$):
  for all Agents $A_i \in A$ do
    if $A_i.timeout()$ is True then
      $A_i.action \leftarrow select\_next\_action(A_i,k)$
    end if
    $A_i$.update_pos()
  end for
```

applying actions stored on the traversed edges. Possible interaction detection is dependant of the TPP; in the case of a visibility based pattern this happens when something enters the agents sensing area (Figure 1a).

5.1. Initialization

In the proposed implementation, each agent is assigned a path that is either precalculated or dynamically generated using traditional path finding algorithms [HNR68, vdBFK06]. Although not strictly necessary, this adds control on the agents something that is lacking in most data-driven crowd approaches. During the first few steps of the simulation (Algorithm 1), each agent moves toward its next target recording its TPP at regular intervals. The visibility parameters such as sampling rate, field of view θ, maximum search radius r, etc., are the same as those used during the construction of the PAG. Instead of updating each agent at every simulation step, agents are distributed into X *bins* and agents in the same bin are updated synchronously. Bins are updated periodically after X frames of simulation time (X corresponds to the trajectory lengths stored on the PAG edges) so that at each simulation step one bin is updated and agents in the remaining bins follow previously selected actions (trajectory segments).

5.2. Interaction Free States

As long as an agent has an interaction free TPP, it can just move towards the target using the trajectory database (Figure 3). A simplified version of the trajectory segment that was followed by the agent over the past few seconds ($4-5$ $2D$ displacement vectors) is used to query the database to retrieve actions that people in the input data followed over similar trajectory patterns (Figure 9). This trajectory is aligned to the moving direction of the agent (as were the stored trajectories). In order to enhance the quality of searches, displacement towards the goal could be used alongside the simplified trajectory representation. Since goals are not clear in the limited view of the source data, we chose not to include displacement. No noticeable artifacts were observed in the simulations since it is very rare to retrieve actions that do not guide agents towards their goals. Additionally, instead of selecting only the best match, the k-nearest are retrieved ($k \leq 10$) and one that moves agents toward their next goal is selected and used under some selection probability to avoid deadlock situations where the agents stay in one place.

5.3. Agent Interaction

As soon as an agent detects a potential interaction (Algorithm 2), it searches to find the node with the best matching TPP in the PAG using a traditional k-nearest algorithm. To improve on the speed and quality of this search, we store

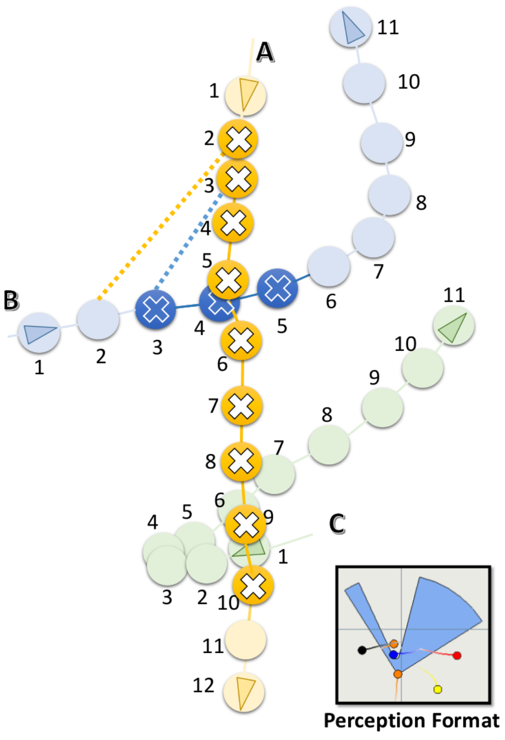


Figure 8: State transitions for three agents (A-C). The numbers next to the agents represent the simulation step and the **X** symbol indicates that the agent is currently in the PAG. A and B sense each other at timesteps 2 and 3 respectively and enter the PAG. B stays on the PAG for three timesteps; up to the moment it resolves the interaction, whereas A keeps using the PAG to steer from both B and C up to timestep 10. C never used the PAG since it never sensed the other two.

simplified (low dimensional) versions of the TPPs in a kd-tree. A good strategy is keeping the lower frequencies of the first and last visibility patterns of the TPPs that when combined encode in essence the relative movement of agents and their neighborhood. A total of 8 to 12 values (4 to 6 coefficients per visibility pattern) were found to be enough to compress a TPP. By indexing into the kd-tree using this representation, the k-nearest matches are found and the best one is selected by comparing the uncompressed TPPs. This TPP indicates the agent's starting node in the PAG to handle the interaction.

Whenever an agent is visiting a PAG node V_i, a series of actions are to be considered depending on the outgoing edges $E_{i,j}$ of the node and any neighboring agents actions. These edges lead to typically different nodes V_j that correspond to different future states that an agent can possibly be in. A fitness function O has been defined (Equation 3) that takes into account neighboring agents future positions and TPPs stored on the different possible future nodes (depending on edge selection). The edge E_k with the minimal value for O is selected as the action to be applied by an agent.

Predicting Future Positions Knowing that the trajectories stored on the edges are of a constant temporal length, an agent traversing the PAG can predict future positions of all neighboring agents depending on their simulation state (either using the IFDB or traversing the PAG (Figure 9)). At any given simulation step, agents can belong either in the currently update bin whereas the remaining ones belong all the other bins. Agents belonging to different simulation bins are already moving on edges since actions were selected for them in previous simulation steps therefore their future positions are well defined and can be found deterministically. For the agents currently being updated, their future positions have to be extrapolated. This is performed using a greedy approach where agents coordinate; given N agents that need to be updated in the current step, an agent A_i is selected (typically the one with fewer possible actions) that predicts the future positions of neighboring agents using their current velocity. After A_i selects an action based on these predictions (see following paragraphs), all remaining $N-1$ agents know that A_i already has selected that action and can therefore calculate deterministically A_i's future position instead of using its current velocity. This procedure continues until every agent is assigned an action. A more expensive alternative approach is to test all possible combinations of actions from all N agents and keep the best ones.

Selecting an Edge Given a set of future positions for agents in a given neighborhood around an agent A_k, its visibility patterns for both the current and next timesteps can be estimated. This set of visibility patterns represent a potential future TPP (P_f); the one that A_k *should* observe after traversing an edge. Having a set of different future TPPs equal to the number of possible actions that A_k can perform, the edge $E_{i,j}$ that minimizes the difference between P_f and the TPP stored in node V_j is selected ($P_n^{(j)}$). In essence, each agent tries to be consistent with the information on the graph; whatever the agent perceives from the environment and represents its state should match what the agent is observing on the PAG graph.

Instead of only trying to minimize the error between the TPPs, an objective function O was defined that takes into account the target distance also:

$$O(P_n^{(j)}, P_f, T_c, T_f) = (1 - corr(P_n^{(j)}, P_f))^{\alpha} \left(\frac{T_f}{T_c}\right)^{\beta} \qquad (3)$$

P_f and $P_n^{(j)}$ represent the predicted TPP and the TPP stored at the end of an edge $E_{i,j}$ respectively, whereas T_f and T_c represent future and current distances from the agent's target. Function $corr(P_f, P_n^{(j)})$ calculates the correlation between the two parameters, with values of 1 and 0 indicat-

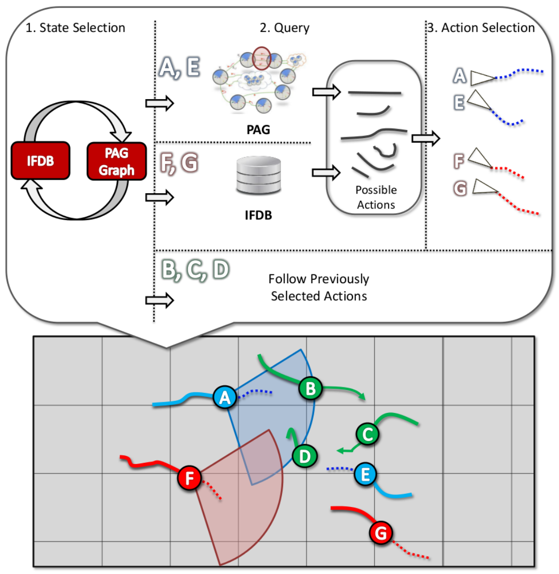


Figure 9: Action Selection Mechanism. Agents B, C and D selected an action in previous simulation steps, whereas the rest need to select an action. Do do so, A and E use the PAG whereas F and G use the IFDB since they currently do not sense any interactions in their FOV. A and E coordinate to select the best possible actions using known trajectories of B, C, D, F and G and extrapolated information.

ing maximum and minimum correlation respectively (Equation 2). Parameters α and β represent weights for the two factors. For most of the presented experiments, weights α and β are assigned to 1; i.e., both are of equal importance.

Handling TPP Errors After traversing an edge, there is a possibility for error between the observed TPP and the ones on currently visiting nodes. This can lead to inconsistencies on the graph walks, i.e., an agent might drift and move on nodes that differ significantly from what the agent perceives. This can happen due to limitations of either the data or the scenarios: either the input data do not have situations similar to ones found in simulated scenarios and therefore not all possible states could be captured, or the graph does not have much connectivity due to limited data and therefore a limited number of actions can be considered. Two policies were considered to handle this based on an error threshold:

- *jump*: when the observed error is large, agents can jump to better matching nodes using the same approach as in Section 5 introducing a penalty in performance at the cost of better quality or
- *no jump*: agents simply ignore the errors and continue traversing the graph trying to minimize the error in the long run, keeping the cost low at the expense of quality (such as collisions, or actions that are not necessarily the best for the given situation).

Algorithm 2 PAG: Select Next Action

```
INPUT
$A_i$: An agent that can be queried for current state.
$k$: Number of nearest neighbors
select_next_action($A_i$, $k$):
  $error \leftarrow 0$
  $action \leftarrow new\_action()$
  $state \leftarrow A_i.query\_environment()$
  if $state.free()$ is True then
    $action.trajectory \leftarrow get\_trajectory\_to\_target(A_i)$
  end if
  if $state.moving\_on\_graph()$ is True then
    $A_i.node \leftarrow A_i.edge.target\_node$
    $error \leftarrow A_i.node.visibility - state.visibility$
  end if
  if error is large or $state.on\_graph()$ is False then
    $A_i.node \leftarrow select\_best\_node(A_i, k)$
  end if
  $A_i.edge \leftarrow select\_best\_edge(A_i.node)$
  $action.trajectory \leftarrow transform(A_i.edge.trajectory)$

  return $action$
```

For most of the experiments, assuming that the graph is well connected and the update frequency is high enough (e.g., every 5 simulation frames), the *no jump* strategy works well, since agents in inconsistent nodes adapt and move towards consistent nodes in just a few steps. The *jump* policy can be thought of as a weight factor on selecting between two data-driven approaches: the simple k-nearest one and the PAG. An *always jump* policy that forces jumps at every simulation step is basically the simple database approach where at each timestep a search is committed to find the best matching state.

6. Results

We have run a number of tests using as input both real and synthetic data (Table 1), to evaluate various aspects of the PAG simulation framework. Additionally, the simulator was implemented employing a server based approach to allow for wide system integration and to decouple the simulation from rendering (assuming a fast and reliable network). The system was implemented in Python and C/C++ with real-time OpenGL and Unity3D rendering clients. All simulations are single threaded and executed on a single core of an Intel Core 2 Quad Core Q8300 clocked at 2.50GHz PC with 4GB of RAM and no GPU acceleration. Evaluation was performed for both simulation quality and performance.

Quality To evaluate the simulation quality, two approaches were followed. On one hand different scenario cases were run using different input data (such as videos of people walking in a busy street, people chatting, students in university

Name	Trajectories	Input Size (frames)	Construction Threshold
lerner-flock	23	766	0.95
lerner-zara	148	9013	0.95
lerner-students	330	4432	0.98
eth-hotel	280	18060	0.98
eth-eth	360	12380	0.98
lee-chat	10	602	0.95
reynolds-pedestrians	50	500	0.99

Table 1: Input datasets used for the experiments in this work.

campus, etc.) from different videos and existing rule based crowd simulation systems and the simulated result was visually compared to the input data (Table 1). On the other hand, different metrics such as speed, angular change histograms and various other metrics were calculated for both the input data and the simulations and then compared (similar to the work of Singh et al. [SNK*08]). The first "looks good" approach is employed by most of the previous works on crowd evaluation and can act as a first indication of the quality of the simulations, whereas the second approach tries to correlate the input and output statistics in a quantitative manner. This two-fold strategy was followed since we believe that crowd simulations should be evaluated both quantitatively and qualitatively; people tend to expect more when evaluating virtual people and in a lot of the cases they might overestimate or underestimate the quality of the underlying algorithms, whereas statistics alone might be misleading since simulations with similar statistics might be completely different in appearance.

Performance A method similar to the one used by Lerner et al. [LCL07] was implemented to study the timing benefit of using the PAG approach. In this method, during simulation and at every simulation step, all of the agents query a database of examples using the agent's current state using Approximate Nearest Neighbors (ANN). Each query returns the k-nearest (state, action) pairs out of which the best example is selected using a more aggressive and accurate comparison method. In order to have a fairer comparison and eliminate any variations that might arise due to different state representations, TPPs were also used to represent state and replaced the state representation described by Lerner et al. [LCL07]. It should be noted here that the state representation by Lerner et al. could be used to generate the PAG graph; TPPs can be any time varying state representation of constant dimensionality (Figure 1). The same coding optimizations and parameters are used throughout the implementation of both methods, thus in the results that follow we emphasize more the relative speed-up and pay less attention to absolute frame rates – a pure C/C++, more optimized, implementation of the same methods would surely be significantly faster.

For all the experiments described in this section, each scenario was run multiple times (10 times) and results were averaged out to reduce artifacts due to noise such as OS task scheduling, disk operations, etc. It should be noted here that in most cases each separate simulation is actually well behaved with low variance (i.e., it is of $\Theta(g(n))$ complexity).

6.1. Preprocessing

Typical graph construction time ranges from 1 to 60 minutes and is highly dependent on the input data size. Construction time includes the data preprocessing and analysis, similarity comparisons and graph generation which is simultaneously stored on disk for later use alongside with all the necessary parameters. A large PAG graph takes up to 30MB when loaded in memory. During our experiments it was found that good values for graph construction are of 95-99% similarity which provide good connectivity and also keep the number of edges relatively small. Typically the similarity threshold is selected to generate on average ≈ 10 edges per node with small variance. This ensures for a fairly good number of alternate choices during graph walks and relatively constant run-time per simulation step. Also, 8-12 DCT coefficients were used for most of the experiments since they keep the most important features of the TPPs (Figure 7).

6.2. Scenarios

A series of different scenarios were used to demonstrate the adaptability of the algorithm on the data and on new environments. Simulated trajectories and timing results for agents with simple targets can be seen in Figure 10. Please consult the accompanying video for the full simulated results.

Opposing Agents Separate experiments with two and four agents moving against each other to reach at the other agent's starting positions were conducted to demonstrate simple interactions. All the experiments were run using different input data (Figure 10) with the exact same parameters; a field of view of 90 degrees was used, with a history of 0.6 seconds and graph construction quality 90-99%. Different source data have a profound effect on the simulations; by using as input a source video with people chatting (*lee-chat*), in both cases the simulated agents stop when they sense each other and start chatting. After a few seconds, they disengage and move towards their targets. By using as input videos of people in busy streets (*eth-hotel, lerner-zara*) the agents avoid each other to reach their targets. The *eth-hotel* dataset has faster pedestrians with larger variations in their moving patterns than the *lerner-zara* dataset and therefore the agents reach their targets faster (21s vs 26s travel time respectively for the four agents) but avoid each other whilst having larger interpersonal distances.

Butterfly To demonstrate once again the subtle differences between the *lerner-zara* and *eth-hotel* datasets of pedestrians a butterfly like environment was simulated. The *eth-hotel* dataset results in a more varied simulation with

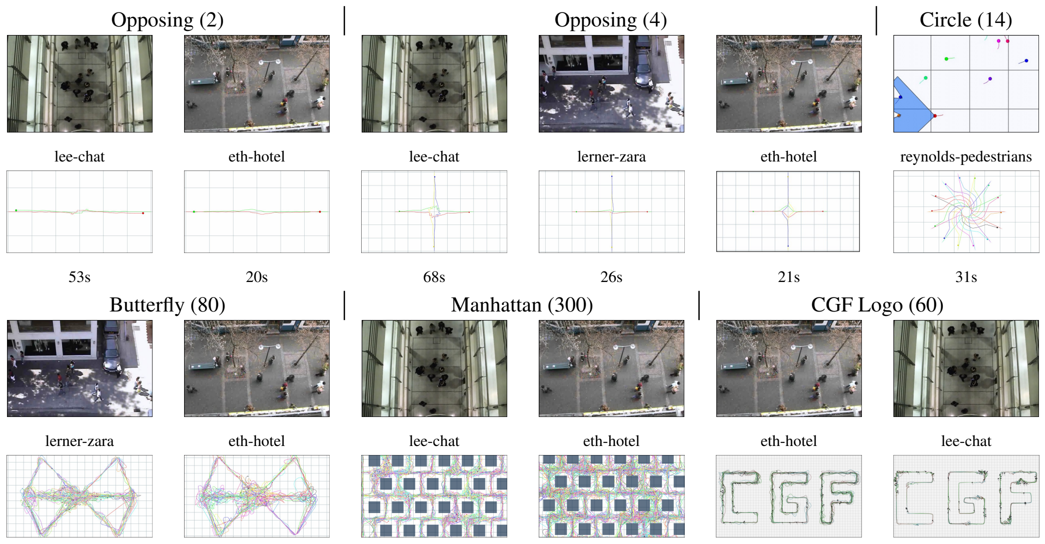


Figure 10: Trajectories for various crowd simulations using different input data and scenarios. Please consult the accompanying video for the simulations.

larger distances to the desired path whereas the behavior derived from the *lerner-zara* dataset is more uniform.

Manhattan A relatively large crowd of 300 agents is simulated (8000 frames) in a Manhattan like environment with city blocks and streets. By using the *lee-chat* dataset, the resulting trajectories are closer to each other and "chatting knots" between simulated agents are observed resulting in gossip town. The *eth-hotel* dataset on the other hand results in wider groups of trajectories covering most of the street areas with minimal chatting adapting to the source data – a street outside a hotel with people moving and sometimes stopping to chat with one another.

CGF Logo Agents in these 5000 frame simulations try to stay on paths that form the *CGF* logo whilst at the same time interacting with each other with varying results depending on the input data. The *lee-chat* dataset generated a thinner logo than the *eth-hotel* dataset with chatting areas visible.

6.3. Speed vs Quality

In what follows, a new trajectory segment for each agent is fetched after every 5 simulation frames in a 25 frames/sec simulation environment. In traditional data-driven techniques at each simulation step every agent generates a query describing its current state and searches an example database for the best matching example. Usually databases are stored in a spatial acceleration structure such as a kd-tree which has $O(log_2 n)$ performance, where n is the number of examples in the database. In the proposed method, a kd-tree is also used in a similar way but only at the beginning of an agent's interaction (see Section 5); i.e., whenever an agent senses some stimuli in its sensing area to find the starting node in the PAG. For all the subsequent frames, that agent just traverses the PAG until interactions are resolved. The number k of (state, action) pairs that are returned by the kd-tree query has a big effect on the speed and the quality of the methods since all these examples are used for action selection either by selecting one of them or by combining them.

In the top row of Figure 11 we see the effect of k on timing and collision-avoidance quality for a sparse crowd. Both methods have similar performance for small k, however the ANN approach has much worse collision-avoidance quality than the PAG method. As we increase k, ANN's collision performance improves but at a much higher cost than PAG. In order for the ANN to obtain similar results to PAG in this scenario the frame rate decreases by more than 18 times. We note here, that one possible way of handling collisions in the PAG framework is by using the *jump* policy described at the end of Section 5 which can act as an intermediate approach to PAG and ANN – given a large enough error on the state, we search for a new node to handle the interaction. Given enough data so that the graph is well connected and the input state space has larger coverage, the PAG typically has better collision-avoidance quality than ANN. Another alternative for collision handling is using a different graph (or database in the case of the ANN) that potentially handles states that the original data could not handle. In the right part of Figure 11 we see the simulation scenario that corresponds to these results. The input trajectories were extracted from a Reynolds Opensteer pedestrian simulation [Rey99] using a similar hexagonal configuration (Table 1).

In Figure 12, the results for a single street scenario can be seen for different input data. The PAG based approach scales

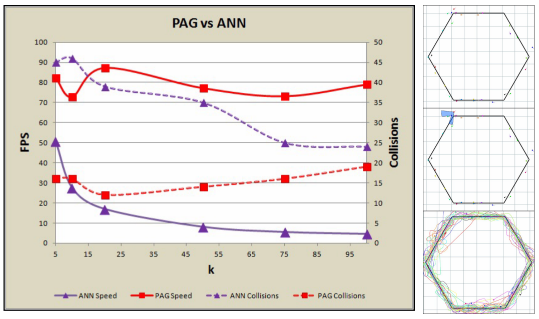


Figure 11: Comparison between the PAG approach and an ANN based one for 2000 frames of simulation time for a hex like environment of 30 agents. The PAG approach outperforms ANN up to 20 times (for $k = 100$) and is virtually independent of the kNN search (lines). It also has much better collision performance (dotted lines).

much better than the ANN approach where performance falls exponentially whereas in the graph based method there is some performance penalty due to the increase of k, but this effect is linear due to the constant number of choices (edges) the algorithm has to consider after selecting the best match. In the ANN approach on the other hand, k results have to be processed at every simulation frame for every agent making this approach very expensive for a real-time environment.

6.4. Simulation Step

In order to have a smooth result the update phase of the simulation needs to be small. As the agents compute a new trajectory segment more regularly, they become more responsive to changes that happen around them and they have smaller deviations from their paths (Figure 13). However, in a traditional data-driven simulation there is a fixed cost per query and as the frequency of the simulation increases, the overall cost increases linearly. In the PAG method the cost of graph traversal is usually much smaller than the initial query to enter the tree; the performance cost comes from the edge selection mechanism and not nearest neighbor searches which are quite slow. Increasing the size of the input data might increase the average edge count, but this happens in a much slower rate and cost increases by a much smaller rate. The input for the presented example was the *lerner-zara* dataset and the simulation environment was a similar sized rectangular area with 30 agents entering from locations similar in position as the input data.

7. Discussion

A new data-driven method for efficient agent-based crowd simulation was introduced, the Perception-Action Graph. Groups of similar Temporal Perception Patterns are identified and grouped together into a graph which in turn is used by simulated agents at run-time. These TPPs are encoded using a compressed form based on their DCT transform to reduce noise and speed up searching for PAG nodes. Agents then perform constrained walks on the graph and try to imitate the behavior of real crowds. Increased performance gains are shown over an approach based on Nearest Neighbors search – the basis for most data-driven crowd simulation algorithms. This method can be used for real-time simulation of data-driven crowds in contrast to most data-driven methods.

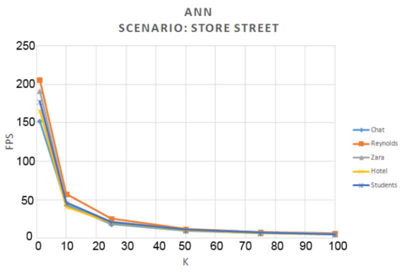
One of the limitations of the current system is that if the simulated scenario is very different from the input data, for example the crowd is much more dense, undesired behaviors such as collisions might appear. Related to this is the amount of data that is really needed to simulate a crowd. It has been observed in our experiments that in a lot of cases increasing the data beyond a certain point did not improve the performance or quality much. In addition, in cases where the graph connectivity is low, the selection of the initial node for entering the PAG becomes even more important. Instead of selecting the best matching node a reward function could be defined that estimates the potential gain of selecting a node over another. One strategy that could be employed to help in this issue could be the use of the jump threshold; i.e., agents might jump during traversal to another node if a large error between their current state and the one currently observed on the PAG is found.

One of the most important issues we are dealing with, is the amount of data that are really necessary to simulate requested scenarios. We are currently investigating approaches to this problem so that both the state and action spaces are covered as much as possible.

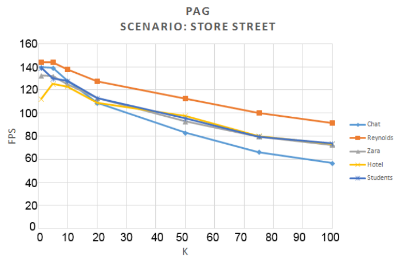
Another limitation of the system is that it does not simulate groups explicitly. Groups do form due to temporal circumstances but might not stay consistent throughout the simulation. Group formation could be modeled in the PAG framework by studying their distribution in the input data and then modifying the optimization function (Equation 3) to take them into account. For example, agents could optimize also for interpersonal relations such as velocity differences or formations similar to Mousaïd et al. [MPG*10]. One alternative solution is to define a higher level controller to constrain the choices of the steering layer. This higher layer controller could be either data-driven, i.e., a formation model is extracted from the data or it could be manually defined.

Currently, goals are not encoded in the state space (TPPs and the trajectory segments of the IFDB). One possible direction, is deriving from the source data goals and integrating them in the state. This approach could potentially increase the quality of the search and simulation.

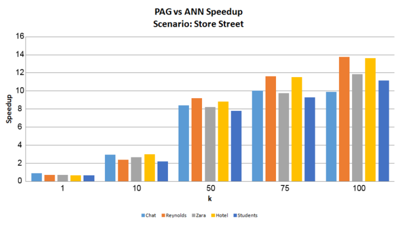
Finally, we are planning on performing some perceptual studies of the quality of the simulations alongside a



(a) ANN

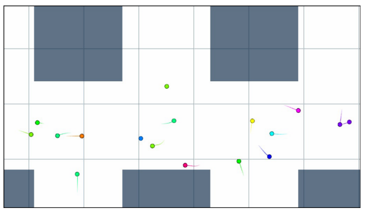


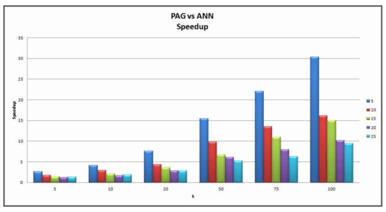
(b) PAG



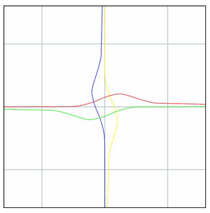
(c) Speed-up

Figure 12: Timing results for a store street scenario having 40 agents for different input datasets. The simulation frame rate for (a) the ANN approach drops exponentially and far below the real-time mark for $k \geq 25$, whereas for (b) the PAG approach decreases linearly but still remains over the real-time mark even for values of $k > 100$.


(a)



(b)


(c) Period = 5 frames

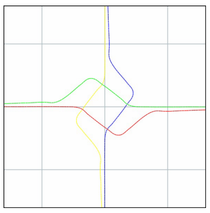

(d) Period = 10 frames

Figure 13: (a) The simulation environment. (b) Increasing the period over which the simulation is updated, the PAG method outperforms ANN by a large margin. (c) (d) Increasing the period from 5 to 10 frames generates a more unresponsive simulation with slightly different behavior (Opposing Scenario trajectories) – the agents sense interactions more or less at the same time but because they use different future predictions (5 against 10 frames) the resulting simulation is different.

more thorough statistical based evaluation since collision-avoidance quality by itself is not sufficient and can be misleading (i.e., a simulation with no collisions at all could simply be agents that move in opposite directions).

Acknowledgments

This work was partially funded by the Cyprus Research Promotion Foundation and the European Structural Funds under contract $AN\Theta P\Omega\Pi I\Sigma TIKE\Sigma/AN\Theta P\Omega/03011(BIE)/08$.

References

[ANR74] AHMED N., NATARAJAN T., RAO K.: Discrete cosine transform. *IEEE Transactions on Computers* (1974), 90–93. 5

[CC07] COURTY N., CORPETTI T.: Crowd motion capture. *Computer Animation and Virtual Worlds 18*, 4-5 (2007), 361–370. 2

[DCOY03] DRORI I., COHEN-OR D., YESHURUN H.: Fragment-based image completion. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 303–312. 2

[HCH10] HILDENBRANDT H., CARERE C., HEMELRIJK C.: Self-organized aerial displays of thousands of starlings: a model. *Behavioral Ecology 21*, 6 (2010), 1349. 2

[HJO*01] HERTZMANN A., JACOBS C. E., OLIVER N., CURLESS B., SALESIN D. H.: Image analogies. In *SIGGRAPH '01: Proc. of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 327–340. 2

[HLTC03] HEIGEAS L., LUCIANI A., THOLLOT J., CASTAGNÉ N.: A physically-based particle model of emergent crowd behaviors. In *Graphicon* (2003). 2

[HM95] HELBING D., MOLNÁR P.: Social force model for pedestrian dynamics. *Physical Review E 51*, 5 (May 1995), 4282–4286. 2

[HNR68] HART P., NILSSON N., RAPHAEL B.: A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on 4*, 2 (july 1968), 100 –107. 7

[HPP05] HSU E., PULLI K., POPOVIC J.: Style translation for human motion. *ACM Transactions on Graphics 24*, 3 (2005), 1082–1089. 3

[Hug03] HUGHES R. L.: The Flow of Human Crowds. *Annual Review of Fluid Mechanics 35* (2003), 169–182. 2

[ICOL05] IRONI R., COHEN-OR D., LISCHINSKI D.: Colorization by example. In *Rendering Techniques* (2005), pp. 201–210. 2

[JCP*10] JU E., CHOI M., PARK M., LEE J., LEE K., TAKAHASHI S.: Morphable crowds. *ACM Transactions on Graphics (TOG) 29*, 6 (2010), 140. 2, 3

[KCPS08] KWON T., CHO Y.-S., PARK S. I., SHIN S. Y.: Two-Character Motion Analysis and Synthesis. *IEEE Trans. on Visualization and Computer Graphics 14*, 3 (2008), 707–720. 3

[KGP02] Kovar L., Gleicher M., Pighin F.: Motion graphs. *ACM Transactions on Graphics (TOG) 21*, 3 (2002), 473–482. 2, 3, 5, 6

[KLLT08] Kwon T., Lee K., Lee J., Takahashi S.: Group motion editing. *ACM Transactions on Graphics (TOG) 27*, 3 (2008), 80. 2

[KSE*03] Kwatra V., Schödl A., Essa I., Turk G., Bobick A.: Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 277–286. 2

[LCF05] Lai Y.-C., Chenney S., Fan S.: Group motion graphs. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), pp. 281–290. 2

[LCHL07] Lee K., Choi M., Hong Q., Lee J.: Group behavior from video: a data-driven approach to crowd simulation. *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), 109–118. 2, 3, 5

[LCL07] Lerner A., Chrysanthou Y., Lischinski D.: Crowds by Example. *Computer Graphics Forum 26*, 3 (2007), 655–664. 2, 3, 5, 10

[LD04] Lamarche F., Donikian S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum 23*, 3 (2004), 509–518. 2

[LFCCO09] Lerner A., Fitusi E., Chrysanthou Y., Cohen-Or D.: Fitting behaviors to pedestrian simulations. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), ACM, pp. 199–208. 2

[Mas07] Massive: Massive software, 2007. 2

[MH03] Metoyer R. A., Hodgins J. K.: Reactive pedestrian path following from examples. In *Proc. of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)* (Washington, DC, 2003), IEEE Comp. Soc., p. 149. 2

[MJBJ06] Musse S. R., Jung C. R., Braun A., Junior J. J.: Simulating the motion of virtual agents based on examples. In *ACM/EG Symposium on Computer Animation, Short Papers* (Vienna, Austria, 2006). 2

[MPG*10] Moussaïd M., Perozo N., Garnier S., Helbing D., Theraulaz G.: The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PLoS One 5*, 4 (2010), e10047. 2, 12

[NGCL09] Narain R., Golas A., Curtis S., Lin M.: Aggregate dynamics for dense crowd simulation. *ACM Transactions on Graphics (TOG) 28*, 5 (2009), 122. 2

[Nør91] Nørretranders T.: *The user illusion: Cutting consciousness down to size.* Viking, 1991. 4

[OPOD10] Ondřej J., Pettré J., Olivier A., Donikian S.: A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 123. 3, 4

[PAB08] Pelechano N., Allbeck J., Badler N.: *Virtual Crowds: Methods, Simulation, and Control*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, 2008. 2

[PdC06] Parseval des Chénes M. A.: Mémoire sur les séries et sur l'intégration complète d'une équation aux différences partielles linéaire du second ordre, à coefficients constants. *Mémoires présentés par divers savants 1* (1806), 638–648. 6

[POO*09] Pettré J., Ondřej J., Olivier A., Cretual A., Donikian S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2009), ACM, pp. 189–198. 2

[POSB05] Pelechano N., O'Brien K., Silverman B., Badler N.: Crowd simulation incorporating agent psychological models, roles and communication. In *First International Workshop on Crowd Simulation, (V-CROWDS '05).* (November 2005), pp. 24–25. 2

[PP10] Pejsa T., Pandzic I.: State of the art in example-based motion synthesis for virtual characters in interactive applications. *Computer Graphics Forum 29*, 1 (2010), 202–226. 2, 3

[PPD07] Paris S., Pettré J., Donikian S.: Pedestrian Reactive Navigation for Crowd Simulation: a Predictive Approach. *Computer Graphics Forum 26*, 3 (2007), 665–674. 2

[Rey87] Reynolds C. W.: Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph. 21*, 4 (Aug. 1987), 25–34. 2

[Rey99] Reynolds C.: Steering behaviors for autonomous characters. *Game Developers Conference 1999* (1999). 2, 11

[RSH*05] Ren L., Shakhnarovich G., Hodgins J., Pfister H., Viola P.: Learning silhouette features for control of human motion. *ACM Transactions on Graphics (TOG) 24*, 4 (2005), 1303–1331. 3

[SACO04] Sharf A., Alexa M., Cohen-Or D.: Context-based surface completion. *ACM Transactions on Graphics (TOG) 23*, 3 (2004), 878–887. 2

[SCCOL06] Schnitman Y., Caspi Y., Cohen-Or D., Lischinski D.: Inducing semantic segmentation from an example. In *ACCV (2)* (2006), pp. 373–384. 2

[SKY08] Shum H. P. H., Komura T., Yamazaki S.: Simulating interactions of avatars in high dimensional state space. *Proceedings of the 2008 symposium on Interactive 3D graphics and games - SI3D '08* (2008), 131. 3

[SNK*08] Singh S., Naik M., Kapadia M., Faloutsos P., Reinman G.: Watch Out! A Framework for Evaluating Steering Behaviors. In *Motion in Games: First International Workshop, MIG 2008, Utrecht, the Netherlands, June 14-17, 2008, Revised Papers* (2008), Springer-Verlag New York Inc, p. 200. 10

[ST05] Shao W., Terzopoulos D.: Autonomous pedestrians. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM Press, pp. 19–28. 2

[TCP06] Treuille A., Cooper S., Popovic Z.: Continuum crowds. *ACM Transanctions on Graphics (TOG) 25*, 3 (2006), 1160–1168. 2

[TLG11] Torrens P., Li X., Griffin W. A.: Building agent-based walking models by machine-learning on diverse databases of space-time trajectory samples. *Transactions in GIS 15* (2011), 67–94. 5

[TM07] Thalmann D., Musse S. R.: *Crowd Simulation*. Springer, 2007. 2

[vdBFK06] van den Berg J., Ferguson D., Kuffner J.: Anytime path planning and replanning in dynamic environments. In *Robotics and Automation, (ICRA) 2006* (may 2006), pp. 2366 –2371. 7

[ZCM11] Zhang Y., Correa C., Ma K.: Graph-based fire synthesis. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), ACM, pp. 187–194. 2