EUROGRAPHICS 2007 / D. Cohen-Or and P. Slavík
(Guest Editors)

Volume 26 (2007), Number 3

Crowds by Example

Alon Lerner[1], Yiorgos Chrysanthou[2] and Dani Lischinski[3]

[1]Tel Aviv University, Israel
[2]University of Cyprus, Cyprus
[3]Hebrew University of Jerusalem, Israel

Abstract

We present an example-based crowd simulation technique. Most crowd simulation techniques assume that the behavior exhibited by each person in the crowd can be defined by a restricted set of rules. This assumption limits the behavioral complexity of the simulated agents. By learning from real-world examples, our autonomous agents display complex natural behaviors that are often missing in crowd simulations. Examples are created from tracked video segments of real pedestrian crowds. During a simulation, autonomous agents search for examples that closely match the situation that they are facing. Trajectories taken by real people in similar situations, are copied to the simulated agents, resulting in seemingly natural behaviors.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

Crowds are an important part of our daily lives. On the streets, at our workplace, or in a shopping mall, we are surrounded by, and become a part of a crowd. The presence and the dynamics of a crowd greatly affect the ambience of a scene. Computer generated crowds are thus becoming common in films, computer games and other virtual world applications and simulations. As these applications continue to strive towards higher levels of realism and scene complexity, there is an increasing need for realistic and believable crowd simulations. Although methods for animating a single human character advance in leaps and bounds, automatically animating a believable pedestrian crowd remains a challenge.

The definition of a "crowd" is very broad, encompassing anything from a battling army to children playing in a schoolyard; however, the collective dynamics exhibited by these crowds differ greatly from each other. Most existing techniques make simplifying assumptions regarding the behavior of the simulated individuals, while still striving to convey the complexity of a real human crowd. While such approaches might capture the broad, overall behavior of the crowd, they often miss the subtle details displayed by the individuals. The range of individual behaviors that may be observed in a real crowd is typically too complex for a simple behavioral model. In this work we explore a data-driven approach instead, where we simulate a pedestrian crowd based on pre-captured examples.

Reynolds [Rey99] divides the behavior of an autonomous agent into three layers: action selection (strategy, goals planning), steering (path determination) and locomotion (animation, articulation). In this paper we focus on the steering aspects of the agents' behavior: our goal is to synthesize a realistic trajectory for each individual in the crowd, such that when these trajectories are combined, a believable crowd behavior emerges. We assume that a separate dedicated mechanism is available for executing the locomotion of each individual. Also, in this work, the agents are not attempting to achieve a specific goal. Rather, they move about in a manner determined by the example input that drives the simulation.

Our approach is data-driven. Given a set of trajectories extracted from a video of a real crowd, we construct a database of examples: each example describes a local spatio-temporal scenario present in the original data. During a simulation we synthesize individual trajectories for each one of the agents in the simulation. Each agent is autonomous, but its trajectory is synthesized incrementally by considering its spatio-temporal relationships with other nearby agents and obstacles, and searching for similar scenarios in the database.

Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

Figure 1: *A typical image of the resulting simulation. A variety of behaviors such as (a) pause to look at the shop window, (b) smooth collision avoidance and (c) group behavior, are implicitly generated by the input examples.*

One way to look at our approach is that we replace a rule-based reactive behavior model with an example-based reactive one. The behavior of our agents is determined as a reaction to the current environment configuration. Rule-based methods typically consider only a small set of predefined behaviors and predefined situations, and account for a limited number of factors in the scene. In our approach, there are as many different situations and reactions as there are examples. Each reaction may depend on an arbitrary number of factors. Thus, in our approach there is a large number of possibly complex rules, implicitly defined by the input data.

In summary, the characteristics of our approach are:

- It produces a realistic crowd behavior with a wide variety of individual agent behaviors (Figure 1).
- It makes no assumptions on the behavior of the agents, other than the set of local spatio-temporal factors used to construct the examples in the database.
- The same system may be used for many different types of crowds and settings by changing the input examples.
- Examples extracted from different situations may be mixed in the synthesis process in order to support different types of agents sharing the same environment.

2. Related Work

Research in crowd simulation has been an active theme in a number of fields recently, such as computer graphics, civil engineering, physics, sociology and robotics. Some popular approaches take ideas from fluid mechanics [Hug03, TCP06], while others make use of particles and social forces [HM95, HLTC03]. Although some of these methods are quite efficient and can capture the aggregate behavior of a crowd fairly well, they cannot fully capture the range or the subtleties of individual behaviors. Simple things such as walking in pairs, stopping to talk to someone, changing one's mind and heading off in a different direction or aimlessly wandering about, are just a few examples which are difficult to capture.

Rule-based methods, on the other hand, can potentially produce realistic results for specific situations. The downside is that they require many finely tuned specific rules which are difficult to define and often go wrong. Given a new situation, a new set of rules needs to be devised. Reynolds [Rey87] proposed one of the earliest rule-based approach that was capable of creating seemingly natural flocking behaviors for animal "crowds". This was later expanded to include additional behaviors such as *seek, pursue* and *evade* [Rey99]. Several works took into account the cognitive decision making mechanism for rule definitions. In the work of Terzopoulos et. al. [TTG94] a range of individual traits, such as *hunger* and *fear*, were defined for simulated fish, generating appropriate behaviors. Funge et. al. [FTT99] simulated agents that not only perceived the environment but also learned from it and use domain knowledge to choose the most suitable behavior out of a predefined set. Musse et. al. [MT97] took into account sociological aspects while defining the behavioral model.

Another strand of the rule-based systems concentrates more on path-planning and steering behavior, [LMM03, LD04, ST05]. In these systems local reactive behavior rules yield different behaviors such as collision avoidance, lane formation and even some limited group behavior. Sung et. al. [SGC04] used a probabilistic approach to choose actions for the simulated agents, while both Farenc et. al. [FBT99] and Thomas and Donikian [TD00] employ information stored in the environment in order to provide behavioral hints to the simulated agents.

Some of the most impressive crowd simulation systems are found in the entertainment industry, for example Massive software [Mas06] and AI.implant [AI.06]. Although natural looking crowds can be generated with these systems, some skill and expertise are required to do so.

The goal of our work is to generate crowds that not only project a natural crowd ambience, but also display a wide range of complex individual behaviors, which may be seen when the simulated agents are examined individually. Our intent is to achieve this goal without defining an explicit behavioral model. Thus, we turn to a data-driven approach, whereby our agents "learn" how to behave from real-world examples.

In recent years example-based 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]. Additional applications include surface completion [SACO04], image colorization [ICOL05], image segmentation [SCCOL06]. Example-based methods have also been successfully used

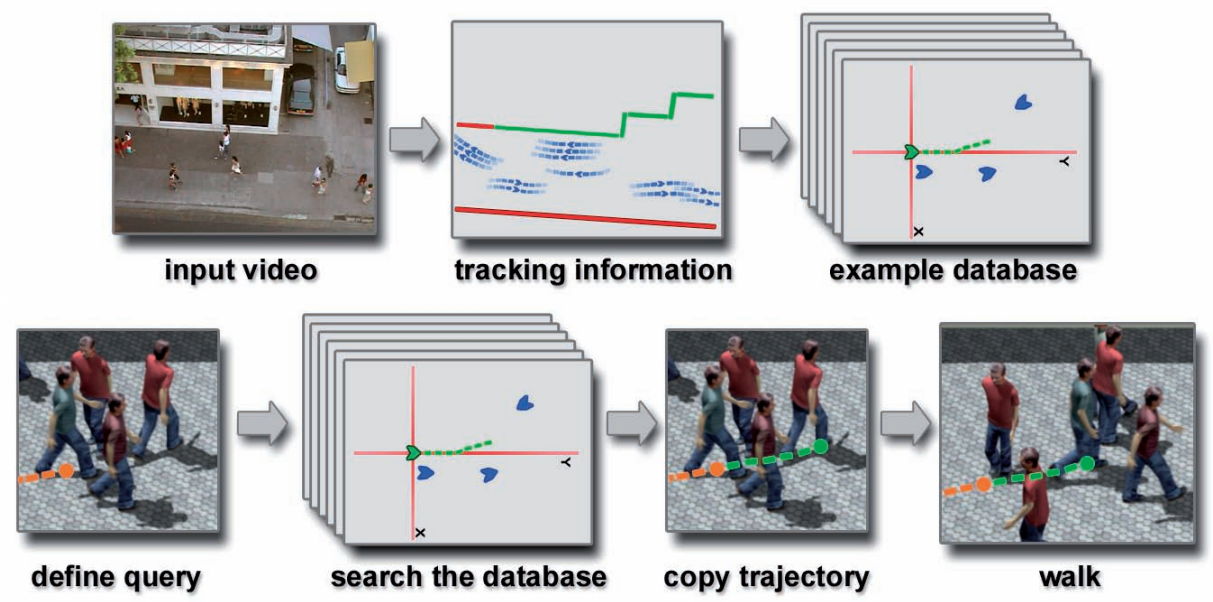


Figure 2: *Overview of our method. The top row depicts the construction of the database, which takes place during pre-processing: the input video is manually tracked generating a set of trajectories. These are encoded as examples and stored in the database. At run-time, bottom row, the trajectories of the agents are synthesized individually by encoding their surroundings (forming a query) and searching the database for a similar example. The trajectory from the example is copied over to the simulated agent.*

in animation and motion synthesis [LT01, KGP02, HPP05, RSH*05].

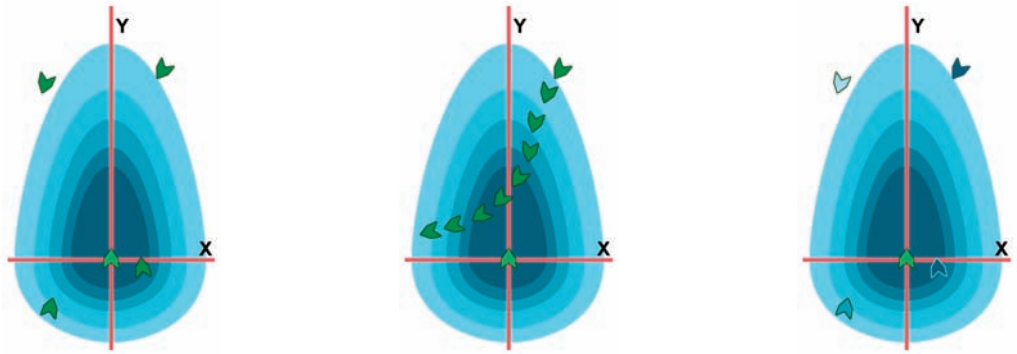
The aforementioned example-based techniques are domain specific and cannot be directly applied to crowd simulation, however there are some crowd simulation works that do use examples. Metoyer and Hodgins [MH03] allow the user to define specific examples of behaviors, while Musse et. al. [MJBJ06] extract paths from a video for a specific environment. In both cases the examples are used to refine an underlying behavior model therefore they are still bound by the limitations of the model. In the work of Lai et. al. [LCF05] a motion graph approach is used for synthesizing group behavior. In order to be able to build a tractable motion graph, the method makes the assumption that the input examples follow a well defined behavior model, such as a flocking system with a restricted configuration space. The great variation in the movements of a human pedestrian crowd would render this method impractical.

3. Overview

In this work we synthesize crowd animations by stitching together pieces of behaviors taken from real-world examples. It is a two part process. As a pre-process we create a database that contains examples. An example is the behavior of one specific person in the given surroundings that influenced that behavior. By behavior we refer to the trajectory of the agent over a short period of time. The second part is the on-line synthesis of the simulated crowd. For each agent we find a similar situation in the database and copy the corresponding trajectory over.

The construction of the example database is described in Section 4 and is illustrated in Figure 2 (top row). The goal is to create a set of examples of how people respond under various conditions. To do that we shot several short (a few minutes long) videos, using a consumer grade camera. The video is shot from an elevated placement in order to avoid the need for ortho-correcting the data and to make the tracking easier. A dedicated tool was developed for manually tracking the people and identifying the static geometry. From each person, as he moves along, we create a number of examples of responses. The crucial question here is: what part of the surrounding environment, moving or static, influences the path of a pedestrian, and what is the extent of this influence? The answer to this question is embodied in our use of *influence functions*, described in Section 4.2.

Once the example database is constructed we can synthesize crowds, as described in Section 5 and illustrated in Figure 2 (bottom row). We start from a static virtual environment with the various parts of the geometry annotated



(a) (b) (c)

Figure 3: *A discrete rendering of the distance-based influence function Inf_E. (a) The surrounding configuration. (b) The trajectory of an influencing agent over the temporal length of a segment. The maximal importance value along the path is used to determine the agents influence value. (c) The surrounding configuration after computing the influence values.*

(e.g., walls and streets are marked as such). The scene is then populated with the desired number of agents, each initialized with a plausible position and velocity. The synthesis process then proceeds as follows. For each agent we construct a *query* that reflects the configuration of his current environment, in a manner analogous to that used in the construction of the examples. The query is used to search for the example with the closest matching configuration. The trajectory followed by the subject of the example is copied over to the simulated agent, generating natural looking behavior. The challenge lies in quantifying the similarity between queries and examples, so as to choose the most appropriate example for each query. This process is described in detail in Section 5.1.

It should be noted that the influence functions described in Section 4.2 and the query matching function described in Section 5.1 are defined in a heuristic manner, based on the intuitive understanding of the behaviors we have observed in the input examples and in real life. Although these functions have performed well in our experiments, it is possible (and even likely) that different heuristics, or functions based on a more systematic study of human behavior might generate even more believable crowds.

4. Building the Example Database

Given a video of a crowd where the trajectories of the different individuals have been tracked and the relevant static geometry in the scene has been identified, our goal is to construct a large *example database*. This database will then be used to generate the trajectories of the agents in the simulated crowd. In this section we describe the construction of the database.

4.1. Examples and Queries

An example, denoted as E in the remainder of this paper, consists of two components. The first component is a segment of the trajectory (path) followed by one of the individuals in the video at some point in time. We refer to this individual as the *example subject*. The second component of an example is the configuration of the factors which might have influenced the trajectory of the subject. An influencing factor might be another individual or some dynamic or static geometry in the scene. For simplicity, we will refer only to agents as potentially influencing factors, however, obstacles are treated in the exact same manner.

For each influencing factor, we store its path (if it's an agent) or its end points and type (for any other object). In either case we also store the amount of influence that we believe it has on the trajectory of the subject. All paths are of a predefined length (set to 40 frames in our simulation) and can be stored either in the compact form of a continuous 2D spline or as a triple of {position, direction, speed} per frame, which is more verbose but faster to use.

Each example is defined using a local coordinate system of its own, with the example subject placed at the origin (0,0) and facing down the positive Y-axis (see Figure 3). All of the influencing factors (other agents and geometry) and their corresponding paths are expressed in this coordinate system.

During the simulation process, whenever we need to determine the next trajectory segment for one of the agents, we form a *query*. Queries are formed in the same manner as examples. The only difference is that when forming a query we might not have all of the future paths of the influencing agents, and they are therefore generated by extrapolating their known paths (up to the time of the query).

4.2. Influence Functions

The trajectory of a pedestrian in a crowd is influenced by numerous factors, which may include the topography of the terrain, the presence of obstacles, proximity to other people, the pedestrian's personality and state of mind, to name a few. Only some of these factors are apparent from a video of a crowd. They are the static geometry (obstacles) in the scene, and the positions and velocities of the individuals in the crowd at different points in time. Thus, the basic simplifying assumption underlying our approach is that those are the only influencing factors that will be accounted for in our crowd simulations. We also assume that our crowds move on an essentially planar terrain.

Consider an agent i at time step t, and let E denote the example that will be created from it. Each agent j in the environment potentially influences the trajectory of i. A few observations are in order here. The first one is that the influence of agents should decrease with distance from i. Next, we observe that agents in front of i are likely to have more influence on its trajectory than those behind i. Finally, we observe that the influence of each agent must be evaluated over a time window, rather than just at a single time instance.

Let the function $Inf_E(j,t')$ denote the *distance-based influence* of agent j on agent i at time t'. We define this distance-based influence function over a local coordinate system aligned with the walking direction of i, as shown in Figure 3. The function itself is given by a product of a Gaussian falloff function, $InfP_E$, in the direction perpendicular to the walking direction and another (asymmetric) falloff function aligned with the walking direction (see Figure 3). Specifically, let d be the (scaled) distance of j from i at time t', then we set

$$InfP_E(j,t') = \exp(\frac{-0.5d^2}{2/v})$$

$$InfF_E(j,t') = \exp(\frac{-0.5d^2}{v})$$

$$InfB_E(j,t') = \exp(\frac{-0.5d^2}{1/2v}),$$

where v is the walking speed of i at time t'. The distance-based influence function is given by the product of these falloff functions:

$$Inf_E(j,t') = InfP_E(j,t') \cdot \begin{cases} InfF_E(j,t') & j \text{ in front of } i \\ InfB_E(j,t') & j \text{ behind } i \end{cases}$$

In order to obtain the *influence function*, $Inf_E(j)$, over a finite time window, we simply use the maximal value attained by $Inf_E(j,t')$ in that time window, normalized by that of all the influencing factors:

$$Inf_E(j) = \frac{\max_{t'} Inf_E(j,t')}{\sum_k (\max_{t'} Inf_E(k,t'))}.$$

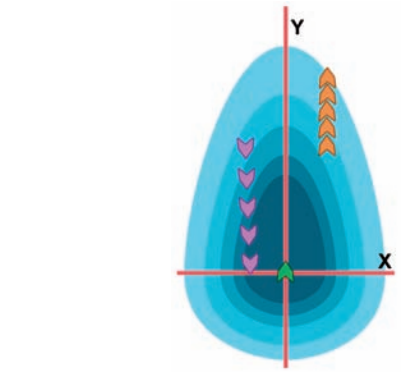


Figure 4: *The influencing agents have similar initial positions relative to the subject. However, due to their different relative velocities, the purple agent (left) has more influence on the subject's eventual path.*

Taking the maximum over the finite time windows is necessary, since it enables us to keep the influence function simple, but at the same time to assign very different influence values to nearby agents having at t similar relative positions, but different relative velocities, with respect to i, as shown in Figure 4. If the maximal influence value is below a predefined cutoff value then we assume that the agent has negligible influence on i's trajectory.

When forming a query, we do not necessarily have all the additional path information for the surrounding agents. If the surrounding agents have simulated path information, we use it in the same manner as in an example. The paths of the subject agent and those of the pathless surrounding agents are extrapolated using their current positions and velocities.

5. Crowd Simulation

Given an example database, a crowd may be simulated. First the scene is populated with the desired number of agents, Then the simulation process repeatedly considers each of the participating agents, and for each one individually decides whether a new trajectory is required. This decision is based on two criteria:

- Has the agent exhausted the last trajectory segment that was assigned to it?
- How significant is the change between the current surrounding configuration and the one that existed at the time of the last assignment?

Note that as the simulation proceeds, the surrounding configuration of an agent continually changes, and influencing factors may enter and leave. If the current configuration differs significantly from the one at the time of the last assignment, the remainder of the segment is discarded and a new trajectory segment is found and assigned. Similarity between

configurations is computed using the matching function described in Section 5.1. In order to preserve smoothness of motion, significant changes between configurations are not checked for the first 15 frames, roughly the duration of one walking step, following a trajectory assignment.

To assign a new trajectory segment to an agent k, we first form a query Q with the agent as the subject, as described in the previous section. Next, we search the example database for the most appropriate example E. Once such an example has been found we assign its path segment to agent k. Similarity values are computed on a continuous scale where values smaller than or equal to 0 represent unrelated configurations, and the value 1 represents the exact same configuration. The temporal length of the trajectory segment copied from E equals the maximal temporal length of a segment times the similarity value between the configurations.

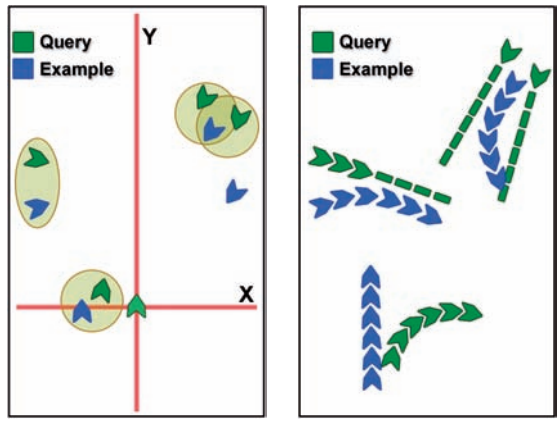
In principle, the most appropriate example to use should be the one with the highest similarity value to the query. However, since the matching is almost never exact, an example with a high similarity value may lead to a collision. Collisions are avoided by examining the paths (either assigned or extrapolated) of each of the influencing factors and testing them for collisions against the example path. If the best match causes a collision, we move to the next best one until a collision free match is found. If no such match exists, we look for a collision avoiding path, as described in 5.2.

5.1. Matching Function

Ideally, for any given query Q we should be able to find an example whose subject walks at the same speed and has the same configuration of influencing factors. Unfortunately, this is highly unlikely to occur in practice. For this reason, we define a function, $Sim(Q,E)$, whose aim is to quantify the similarity between a query Q and an example E. In order to generate plausible behaviors, the influencing factors from Q should be matched to those in E, and the affinity of each match assessed (see Figure 5). The affinity between an influencing agent $k \in Q$ and $j \in E$ is measured using their positions, directions and velocities along their path segments in the local coordinate system of the query:

$$Aff(k,j) = \underset{t'}{avg}(SimVal_{t'}(k,j)),$$

where $SimVal_{t'}(k,j)$ is a multiplication of Gaussians, quantifying the similarity between the influencing factors. In the case of matching agents, the similarity is quantified in terms of differences in position, direction and velocity. In the case of matching obstacles, such as walls, the similarity is quantified in terms of the differences in the direction of the obstacles and the positions of the points on the obstacles which are closest to the query subject. The variance of all the Gaussians is inversely proportional to the velocity of the query subject. Note, that a person cannot be matched to an obstacle since their behaviors are fundamentally different.



(a) (b)

Figure 5: *The matching function. (a) The circles represent matches between query and example influencing agents. Not all of the influencing agents are matched in this case. (b) The quality of the match is determined not only according to the current frame, but rather according to the average matching over time. For a query agent, missing path segments are extrapolated.*

An influencing factor $k \in Q$ is matched to the influencing factor $j \in E$ which yields the highest affinity value $M_E(k)$. As a result several influencing factors from the query can be matched to the same influencing factor from the example. Therefore, when calculating the matching value we use $AvgInf_E(j)$ which divides $Inf_E(j)$ by the number of times that j was matched to an influencing factor from Q:

$$M_E(k) = \frac{Inf_Q(k) + AvgInf_E(j)}{2} Aff(k,j)$$

where:

$$\mathrm{j} = \arg\max_{i \in E} Aff(k,i).$$

If not all the influencing factors in the query Q or example E were matched, we compute a penalty value $Um(Q,E)$, which is an average of the query and example penalties. We define the penalty to be the sum of squared unmatched influence values:

$$Um(Q,E) = \frac{1}{2}\left(\sum_{k \in Q_{unmatched}} Inf_Q(k)^2 + \sum_{j \in E_{unmatched}} Inf_E(j)^2\right).$$

In order to assure a smooth transition between one path segment and another, the velocities of the query and example subjects must be similar, therefore we use a Gaussian, $S(Q,E)$, which quantifies the difference in velocities. Its variance is inversely proportional to the velocity of the query subject.

Finally, the similarity function, $Sim(Q,E)$, sums the matched values, subtracts the penalty value, $Um(Q,E)$, and

multiplies the result by the velocity matching function, $S(Q,E)$:

$$Sim(Q,E) = S(Q,E) \cdot (\sum_{k \in Q} M_E(k) - Um(Q,E)).$$

5.2. Collision Avoiding Paths

In the ideal case, collisions should not occur in our simulations. If the configuration space is properly covered, a collision free matching example should always be found. However, due to incomplete coverage, occasionally a situation may arise where none of the matching examples leads to a collision free solution. In such cases we employ our collision avoiding scheme. We search for a path segment that will steer the agent away from the collision, without matching the surrounding configuration. By doing so we essentially define as many collision avoiding rules as there are collision avoiding paths in the database.

The number of different path segments that resolve the collision is large. In order to preserve believability of behavior, we choose one that assures a smooth transition into the chosen segment. To do so we use the affinity function to quantify the similarity between a segment of the query's trajectory history to that of the example trajectory history. We choose the path that maximizes the function while avoiding the collision. If more than one such path exists, one is chosen at random.

6. Discussion

Synthesizing crowds by example requires the creation of an example database. Examples are created from videos of real crowds. Standing on a roof of a building, we shot several crowd videos, each one approximately 5 minutes in length (see Figure 6). Although there has been some progress recently in automatically tracking movement of people in a crowd [BC06], no robust method was available to us. Therefore, we manually tracked the people. Additionally, we marked buildings, walls, doors and other influential objects, such as the edge of the sidewalk. Tracking times varied depending on the length of the video and the number of people that appear in it. Several hours are required for tracking a sparse crowd while a denser one may require a day.

The number of examples that can be created from a video depends on its length and the density of the crowd that appears in it. An example can be created from an individual at a certain frame if tracking information exists for both the individual and its influencing factors over the temporal length of the path segment (set to 40 frames in the results reported here). Obviously, two examples created from the same subject at consecutive frames are likely to be very similar, if not identical. We applied our matching function on examples from consecutive frames, filtering out nearly identical ones. Table 1 shows the number of examples that were created from the different input videos. The time required to construct an example database ranges between 30 seconds for a sparse crowd video to 8.5 minutes for a dense one. The amount of memory required to store it ranges between 50MB to 150MB.

	input crowd		
	sparse	dense	synthetic
# of unfiltered examples	42772	151557	1455
# of filtered examples	18912	81775	612

Table 1: *The number of examples created from the three input crowds, before and after filtering.*

When synthesizing a crowd, the number of times the database is queried determines the amount of time required to synthesize the crowd. Aside from the length of the simulation and the number of people that appear in it, the number of queries is affected by several factors, among them:

- The similarity between the example crowd and the simulated one.
- The variety of examples that exist in the database.
- The presence of obstacles.

The similarity between the example crowd and the simulated one determines the quality of the matches between examples and queries. Synthesizing a dense crowd using a sparse crowd database will likely lead to partial matches at best, leaving influencing people from a query unmatched. As a result the number of times that each simulated person queries the database grows. However, this does not imply that the same number of people should appear in the input and synthesized crowd. As long as examples exist in the input where the local density resembles that of the simulated crowd, then sufficiently similar examples are likely to be found.

The variety of examples in the database influences both the richness of behaviors and the amount of time required to synthesize the crowd. Creating an example database from a crowd that behaves in a uniform fashion, displaying little or no variety of behavior, results in a crowd that behaves in roughly the same manner. To illustrate this point, we created a small synthetic database where people walk at constant speed, in straight lines, do not stop and make either 90 degree or U-turns. The resulting synthesized crowd, which can be seen in the accompanying video, does not display any behaviors more complex than the concatenation of the above mentioned behaviors.

The lack of variety affects the overall running time of the simulation. A simulated person uses the examples to guide it through the crowd, essentially telling it how to behave in

Figure 6: *The sparse (*zara*) and dense (*student-day*) input videos and a screenshot of the simulation.*

a given configuration. If the database does not sufficiently cover the configurations space, then the number of times where a query will not be adequately matched grows, generating more and more queries.

The presence of obstacles also affects the required computational time. When two people are about to collide with each other, there is a mutual understanding that both of them will try to avoid the oncoming collision. Our simulation tries to mimic this behavior. The matching function takes into account the path that each influencing person follows in the upcoming frames, if this path is not available then it is extrapolated. By doing so, when a simulated person queries the database it essentially predicts the paths of its influencing people and searches for an example that matches the predicted paths. Out of the two people about to collide, the first one to query the database predicts the path of the other person. However, the second person knows the exact path of the first one, since he just queried the database. Therefore, his matching example will steer him away from the collision.

The problem with obstacles is that they do not react to people. They do not move away from an oncoming person and do not contribute to collision avoidance. However, if examples avoiding the collision exist in the database, they will eventually be found and the collision will be avoided, but several queries might be required to do so.

7. Results

The size of our example database runs in the tens of thousands of examples, therefore a brute force search examining the entire database for each query is impractical. We employ an Approximate Nearest Neighbor search in order to find the best examples quickly, and limit the number of examples checked for each query. On the other hand, in the current implementation, the search for a collision avoiding path is implemented in a brute force manner. In this non-optimized approach, all the possible paths are checked before one is chosen. Although this approach significantly slows down the simulation, the collision avoiding paths are chosen from a large variety of options and repetitions are avoided.

Our technique requires several parameters to be tuned, such as the maximal length of a trajectory or the scaling factors for the similarity functions. One can view this as an optimization problem in a high dimensional search space. By running different simulations, we tuned each parameter independently, generating a set that produced the desired results.

We synthesized four different crowds using our technique. In all cases the simulations we created are 2 minutes long, running at 25 frames per second, which translates to 3000 synthesized frames. The first three crowds were synthesized using the sparse crowd input video. This video has an average of 5-6 people in each frame and includes features such as walls, a door and the road curb. We extended the environment that appeared in the video adding an additional building. The crowds that were synthesized consist of 2, 8 and 20 people on average per frame. The crowd consisting of only two people runs in real time. The 8 person crowd required about 6 minutes of computation (that is an average of about 8fps for the 3000 frames), while the 20 person crowd requires about 10 minutes (an average of about 5fps).

We synthesized a dense crowd consisting of 40 people on average per frame using the dense crowd input video, which has roughly the same number of people per frame. This simulation required one hour to synthesize the 3000 frames. The long computation time stems from the fact that the 80k examples that appear in the database do not properly cover the configuration space. As a result, the number of times that collision avoiding paths are required grows, slowing down the simulation. In a dense crowd the dimensionality of the configuration space is high, since there are many influencing factors appearing in each query and example. For a dense crowd, 5 minutes of input video are insufficient to generate a proper example database which will accelerate the simulation.

All the simulations are collision free and display plausible behaviors on the individual and group levels. We refer to the accompanying video for clips of the synthesized crowds. Different behaviors are apparent in the crowd; such as changes in walking direction (see Figure 7(top)), smooth

Figure 7: *(Top) The agent in the yellow T-shirt turns and heads in a different direction. (Bottom) Smooth collision avoidance - the agent in the grey T-shirt smoothly navigates between the on coming agents.*

collision avoidance (see Figure 7(bottom)), group behavior and others. In addition the simulated agents learn from the examples how to relate to the various features of the environment. For example, entering a building or standing looking at a shop window (Figure 1). The trajectory of each one of the simulated agents is comprised from the concatenation of path segments taken by real people walking in a real crowd. When a query is matched to an example, its walking speed is matched to the example trajectory. When a trajectory segment is copied over to a simulated agent it is aligned to its walking direction. Therefore, any changes in speed and direction in the simulation are derived from similar occurrences from the example set. Our simulation technique concatenates existing trajectories, and by doing so, believable crowd behavior emerges.

8. Conclusions and Future Work

In this paper we presented a novel crowd simulation technique. Our approach is data-driven, using trajectories extracted from a video of a real crowd to drive the simulated agents. It is an example-based reactive simulation, where agents react to the configuration of their surrounding environment. An example defines a reaction to a certain configuration, therefore implicitly defining a reactive rule or behavior. The variety of behaviors displayed by our agents is limited only by the number of examples.

In our approach no assumptions are made on the behavior of the agents, other than the definition of the spatio-temporal influencing factors. By using a continuous distance-based influence function we assign relative weights to each such factor. Thus, our agents react to the configuration as a whole, rather than to each factor independently. Our influence function is general and can be easily incorporated into rule based simulations. In such simulations the function can be used to determine how to blend between rules reacting to the different influencing factors.

Our system is flexible and can easily simulate different types of crowds. By changing the input examples, different reactive behaviors are introduced, therefore changing the behavior of the simulated crowd. One can incorporate examples from different inputs, allowing for agents of different types to share the same environment. Additionally, the simulation is scalable where the global density of the crowd can be changed, as long as the local density surrounding the agents remains relatively similar to the input examples.

In this work, the simulated agents are not attempting to achieve a specific goal, or reach a certain location. An interesting avenue for further research would be to incorporate into our data-driven simulation a cognitive model, thereby directing the matching function to search for examples that will guide the agents towards their goals. Another interesting avenue for further research would be to find an optimal definition of the influence and matching functions, along with a combination of their parameters.

References

[AI.06] AI.IMPLANT: Ai.implant software, 2006.

[BC06] BROSTOW G., CIPOLLA R.: Unsupervised Bayesian Detection of Independent Motion in Crowds. *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1* (2006), 594–601.

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

[FBT99] FARENC N., BOULIC R., THALMANN D.: An informed environment dedicated to the simulation of vir-

tual humans in urban context. *Computer Graphics Forum 18*, 3 (Sept. 1999), 309–318. ISSN 1067-7055.

[FTT99] FUNGE J., TU X., TERZOPOULOS D.: Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. In *Siggraph 1999, Computer Graphics Proceedings* (Los Angeles, 1999), Rockwood A., (Ed.), Addison Wesley Longman, pp. 29–38.

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

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

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

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

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

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

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 473–482.

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph. 22*, 3 (2003), 277–286.

[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.

[LD04] LAMARCHE F., DONIKIAN S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Comput. Graph. Forum 23*, 3 (2004), 509–518.

[LMM03] LOSCOS C., MARCHAL D., MEYER A.: Intuitive crowd behaviour in dense urban environments using local laws. In *TPCG* (2003), pp. 122–129.

[LT01] LIM I. S., THALMANN D.: A vector-space representation of motion data for example-based motion synthesis. In *DEFORM '00/AVATARS '00: Proceedings of the IFIP TC5/WG5.10 DEFORM'2000 Workshop and AVATARS'2000 Workshop on Deformable Avatars* (Deventer, The Netherlands, The Netherlands, 2001), Kluwer, B.V., pp. 169–179.

[Mas06] MASSIVE: Massive software, 2006.

[MH03] METOYER R. A., HODGINS J. K.: Reactive pedestrian path following from examples. In *CASA '03: Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)* (Washington, DC, USA, 2003), IEEE Computer Society, p. 149.

[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).

[MT97] MUSSE S. R., THALMANN D.: A model of human crowd behavior: Group inter-relationship and collision detection analysis. In *Computer Animation and Simulation* (1997), pp. 39–52.

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

[Rey99] REYNOLDS C.: Steering behaviors for autonomous characters. *Game Developers Conference 1999* (1999).

[RSH*05] REN L., SHAKHNAROVICH G., HODGINS J. K., PFISTER H., VIOLA P.: Learning silhouette features for control of human motion. *ACM Trans. Graph. 24*, 4 (2005), 1303–1331.

[SACO04] SHARF A., ALEXA M., COHEN-OR D.: Context-based surface completion. *ACM Trans. Graph. 23*, 3 (2004), 878–887.

[SCCOL06] SCHNITMAN Y., CASPI Y., COHEN-OR D., LISCHINSKI D.: Inducing semantic segmentation from an example. In *ACCV (2)* (2006), pp. 373–384.

[SGC04] SUNG M., GLEICHER M., CHENNEY S.: Scalable behaviors for crowd simulation. *Comput. Graph. Forum 23*, 3 (2004), 519–528.

[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.

[TCP06] TREUILLE A., COOPER S., POPOVIC Z.: Continuum crowds. *ACM Trans. Graph. 25*, 3 (2006), 1160–1168.

[TD00] THOMAS G., DONIKIAN S.: Modelling virtual cities dedicated to behavioural animation. In *Eurographics 2000* (2000), Blackwell Publishers.

[TTG94] TERZOPOULOS D., TU X., GRZESZCZUK R.: Artificial fishes: autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artif. Life 1*, 4 (1994), 327–351.