Using Computer Vision to Simulate the Motion of Virtual Agents

Soraia R. Musse, Cláudio R. Jung, Julio C. S. Jacques Jr. and Adriana Braun

Graduate Program on Applied Computing

Universidade do Vale do Rio dos Sinos

São Leopoldo, RS, Brazil, 93022-000.

Tel: +55-51-3591-1122 ext. 1626

Fax: +55-51-3590-8162

Abstract

In this paper, we propose a new model to simulate the movement of virtual humans based on trajectories captured automatically from filmed video sequences. These trajectories are grouped into similar classes using an unsupervised clustering algorithm, and an extrapolated velocity field is generated for each class. A physically-based simulator is then used to animate virtual humans, aiming to reproduce the trajectories fed to the algorithm and at the same time avoiding collisions with other agents. The pro-
posed approach provides an automatic way to reproduce the motion of real people in a virtual environment, allowing the user to change the number of simulated agents while keeping the same goals observed in the filmed video.

**Keywords:** crowd simulation, panic situations, computer vision
Introduction

The behavior of virtual humans has been investigated by several research groups, with a variety of purposes. For instance, virtual human modeling can be used in entertainment areas, to simulate convincingly the motion of several virtual agents (e.g. movies and games); to populate immersive virtual environments aiming to improve the sense of presence (e.g. collaborative virtual environments); to supply simulation of virtual human motion for the evaluation of constrained and complex environments (e.g. to simulate the flow of people leaving a football stadium after a match). Also, crowd models can play an important role in order to determine the level of comfort and security of people in large public spaces, as in architecture, urban planning and emergency management.

Several challenges are involved in strategies for simulating the motion of virtual humans. In some works, the behavior of virtual humans is in some way pre-programmed, usually related to constraints of the virtual environment. In general, such techniques illustrate a virtual world “artificially populated” by people that seem to behave in a random way. In other works, the user can observe real life data to calibrate the simulation, and empirically reproduce main directions, attraction points, etc. Such methods may produce realistic simulation results, but demand a considerable amount of user interaction. Yet another class of techniques is focused on the simulation of a large number of virtual agents. However, most of the approaches for crowd simulation are focused on panic situations, and it is still a challenge to develop simulation methods that mimic accurately real life situations (concerning
virtual human motion and behavior for non-panic situations).

This paper presents a new approach for simulating the motion of virtual agents based on examples, using input information captured from real life. As it will be explained later, our system uses the trajectories of tracked people in filmed video sequences to simulate virtual humans in normal life (meaning typically observed situations), being able to extrapolate the number of simulated people while keeping motion characteristics of real filmed people. For instance, if a video sequence contains only ten people, the proposed model can use the trajectories of these individuals and extrapolate the number of virtual people in a simulated environment (e.g. 1000 people), trying to keep the same motion characteristics (such as velocities, spatial occupation, etc.). In fact, video captured data are used as examples to provide more realistic control of virtual humans, as well as to provide information for validation, what is in particular an important challenge in the domain of simulated people motion. Since we capture real life data, we are also able to compare the results of our simulation with filmed sequences in similar conditions.

In summary, the main contribution of this work is to simulate the motion of virtual humans from examples, which are extracted from real video sequences using computer vision algorithms. The proposed technique is fully automatic, and does not require user intervention. A very brief overview of this method was shown in [1], focusing on editable trajectories to provide better control to the animator.


Related Work

Virtual agents and groups have been studied since the early days of behavioral animation. Reynolds [2] simulated flocks of bird-like entities called boids. He obtained realistic animation by using only simple local rules. Tu and Terzopoulos [3] created groups of artificial fishes endowed with synthetic vision and perception of the environment, which control their behavior. Bouvier, Cohen and Najman [4] have studied crowd movements with adapted particle systems. In their work, the motion of people is modeled with charges and decision fields, which were based on the interactions between electric charges and electric fields.

Goldenstein et al. [5] developed a dynamic non-linear model to simulate the movement of a large number of agents in a virtual environment that can be modified in real time. Helbing and collaborators [6, 7] presented a model to simulate groups of people in panic situations. Helbing’s model is physically-based, and crowd movement is determined by attraction and repulsion forces between simulated people and the environment. Braun et al. [8] extended this model by adding individualities to agents and including the concept of groups, focusing on panic situations. More recently, Helbing’s team [9] presented a paper where they discuss possible validations of their results based on empirically observed situations.

Musse and Thalmann [10] proposed a model with hierarchically structured crowds having different levels of autonomy. In their model, the behavior is based on a set of rules dealing with the information contained in the groups of individuals. This work also intro-
duced the KSI methodology, in which agents and groups are represented by their knowledge, status and intentions. More recently, Ulicny and Thalmann [11] proposed a model for crowd simulation based on combination of rules and Finite State Machines for controlling agents’ behaviors in a multi-layer approach.

Metoyer and Hodgins [12] proposed a method for generating reactive path following based on the user’s examples of the desired behavior. In their approach, the gross character motion is specified by the user via desired paths, while the fine details of the navigation (such as obstacle avoidance and path following) are implemented with automatic reactive navigation techniques.

Lai et al. [13] used a data-driven animation technique, namely Group Motion Graphs, to simulate the movement of groups of discrete agents, such as flocks, herds, or small crowds. Pelechano [14] proposed an approach that applies a combination of psychological and geometrical rules layered on top of a social forces model to simulate large and dense crowds of autonomous agents. Treuille et al. [15] presented a real-time crowd model based on continuum dynamics. In their model, a dynamic potential field simultaneously integrates global navigation with moving obstacles.

Some work have also been done to determine the desired velocities for agents in crowd simulation. Brogan and Johnson [16] presented an approach based on empirical observation of real life situations to describe path planning during a suite of experiments. They compared their pedestrian model with the observed path and also with the A* model, by using metrics like distance, area and speed of individual trajectories. The model uses these met-
rics to compare individual trajectories, but determining similarity patterns between groups of people is a different problem. Other methods presented techniques to design velocity fields based on procedural flow generators to describe the motion of a large number of entities. Chenney [17] presented a model for representing and designing velocity fields through a natural user interface. Crowdbrush [18] was presented by Ulicny et al., and aims to provide an authoring tool to describe initial locations, orientations, texture colors, emotions and paths for crowds, in real time. Pettré et. al [19] introduced a framework for real-time simulation and rendering of crowds navigating in a virtual environment. In their work, the 3D environment is automatically structured into a graph to capture its topology, and then a navigation planning method is used to produce a variety of paths.

All techniques described in this Section require some kind of stimulus to guide the motion of virtual agents. Such stimulus (desired velocity or new position) can be generated using several different approaches, such as local rules [2, 10, 11, 13, 14], physically-based systems [4, 6, 8, 15] and user-based interfaces [12, 16–19]. Although several of these techniques may be used to simulate normal life situations, they require the observation of real life and a manual calibration of parameters (stimuli), demanding considerable user interaction. In this paper, we capture the desired velocities automatically using computer vision algorithms and feed such information to a virtual human simulator, without any user intervention.
Automatic People Tracking using Computer Vision

As it will be discussed in Section “The Simulation Model”, our simulation algorithm requires that each agent must have a desired velocity at each position. In this work, we use computer vision techniques to track individuals from existing video sequences, and generate extrapolated velocity fields to feed the simulation algorithm. The tracking algorithm is briefly explained next, and the generation of extrapolated velocity fields is provided in Section “Using Tracked Trajectories to Obtain Desired Velocities”.

Several vision-based techniques for people tracking in video sequences have been proposed in the past years [20–24], most of them with surveillance purposes. In these applications, an oblique (or almost lateral) view of the scene is required, so that faces of the individuals can be recognized. However, such camera setups often results in occlusions, and mapping from image pixels to world coordinates may not be accurate (due to camera projection). Since our main goal is to extract trajectories for each individual, we chose a camera setup that provides a top down view (thus, reducing perspective problems and occlusions).

In fact, the general plane-to-plane projective mapping is given by:

\[
\begin{align*}
x &= \frac{au + bv + c}{gu + hv + i}, \\
y &= \frac{du + ev + f}{gu + hv + i},
\end{align*}
\]

where \((u, v)\) are world coordinates, \((x, y)\) are image coordinates, and \(a, b, c, d, e, f, g, h, i\) are constants. In top down view camera setups, it is easy to show that Equation (1) reduces to:

\[
x = au, \quad y = ev,
\]
and the mapping from image to world coordinates is trivial. Furthermore, Equation (2) indicates that persons have the same dimensions at all positions, indicating that an area thresholding technique may be used for people detection.

The first step of the tracking procedure is to apply the background subtraction algorithm described in [25], which consists of obtaining a mathematical model of the scene background (assuming that the camera is static), and subtracting each frame of the video sequence from this model. Extracted pixels are tested for shadow detection, and the remaining pixels form connected foreground pixels (called blobs).

In the first frame that a new person enters the scene, it is expected only a portion of his/her body to be visible. As the person keeps moving, a larger portion of the body will be visible, until he/she enters completely in the viewing area of the camera. Hence, each new blob that appears is analyzed, and its area is computed in time. If this blob is related to a real person, then its area will increase progressively, until it reaches a certain threshold that represents the minimum area allowed for a valid person (such threshold is determined \textit{a priori}, based on average person sizes in a specific camera setup).

Once a person is fully detected in the scene, it must be tracked across time. Several tracking algorithms that rely on lateral or oblique camera setups explore the expected longitudinal projection of a person to perform the tracking, as [20, 24, 26]. However, such hypothesis clearly does not apply in this work, requiring a different strategy for people tracking.

In top down view (or almost top down view) camera setups, a person’s head is a relatively
invariant feature, indicating that tracking can be performed through template matching. To find the approximate position of the center of the head, the Distance Transform (DT) is applied to the negative of each foreground blob (i.e. background pixels are used as the binary image required to compute the DT). The global maximum of the DT within each foreground blob corresponds to the center of the largest circle that can be inscribed in the blob, and it provides an estimate of the person’s head center. The bounding box of such circle is used as the template $T$ to be matched in subsequent frames. Figure 1(a) illustrates our background subtraction and shadow removal algorithm, along with the selection of the initial correlation template (red square).

The next step is to identify template $T$ in the following frame. Although there are several correlation metrics designed for matching a small template within a larger image, Martin and Crowley’s study [27] indicated that the Sum of Squared Differences (SSD) provides a more stable result than other correlation metrics in generic applications, leading us to choose the SSD as the correlation metric.

Also, it does not make sense to compute the correlation over the entire image, since a person moves with limited speed. We use a reduced correlation space, taking into consideration the frame rate of the sequence, and the maximum speed allowed for each person. For example, if the acquisition rate is 15 frames per second (FPS) and the maximum allowed speed is 5 m/s, than the center of the template cannot be displaced more that $5 \times \frac{1}{15} = 0.33$ meters in the subsequent frame (for the camera setup shown in Figure 1, 0.33 meters correspond to approximately 35 pixels). Since the template center must be moved to a foreground
pixel, the correlation space is further reduced by removing all background pixels. The SSD between the template $T$ and the correlation space is then computed, and the center of the template is moved to the point related to the global minimum of the SSD. Such correlation procedure is repeated for all subsequent frames, until the person disappears from the camera view.

Although the head is a good choice for the correlation template, head tilts and illumination changes may vary the graylevels within the template. Also, the procedure for selecting the initial template may not detect exactly the center of the head. To cope with such situations, $T$ is updated every $N_f$ frames (we used $N_f = 5$ for sequences acquired at 15 FPS). An example of several trajectories tracked using the proposed automatic procedure is illustrated in Figure 1(b).

As a result of our tracking procedure, we can determine the trajectory (hence, the velocity) of each person captured by the camera. Extracted trajectories can be used directly in some simulators (e.g. [12, 16]), but other simulators require a vector field that provides the desired velocity for each person at each point of the image (e.g. [6–8, 17]). In the next Section, we describe our implementation of a virtual human simulator that requires the full velocity field, along with a method for obtaining such velocity field from a (probably sparse) set of tracked trajectories.
The Simulation Model

One advantage of simulation models based on Physics is the possibility of easily including attraction and repulsion forces, that can be used for collision avoidance. One of the physically-based models widely referred in the literature was proposed by Helbing et al. [7], who developed a model based on Physics and Sociopsychological forces in order to describe the behavior of human crowds in panic situations. Such model is in fact a particle system where each particle $i$ of mass $m_i$ has a predefined velocity $v_i^g$ (goal velocity, typically pointing toward exits of the virtual environment) to which it tends to adapt its instantaneous velocity $v_i$ within a certain time interval $\tau_i$. Simultaneously, each particle $i$ tries to keep a velocity-dependent distance from other entities $j$ and walls $w$, controlled by interaction forces $f_{ij}$ and $f_{iw}$, respectively. The change of velocity in time $t$ for each particle $i$ is given by the following dynamical equation:

$$m_i \frac{dv_i}{dt} = m_i \frac{v_i^g - v_i(t)}{\tau_i} + \sum_{j \neq i} f_{ij} + \sum_w f_{iw}$$  \hspace{1cm} (3)

Although Helbing’s equation has been exhaustively used to simulate panic situations, it is unknown to us its usage for “normal life” behaviors, since it is not trivial to obtain the goal velocities $v_i^g$. Next, we describe an algorithm to obtain such velocities from trajectories extracted in video sequences. We also emphasize that we chose Helbing’s model in this work because it would be very easy shift from normal life to panic situations, which is also an important research topic. For panic situations, the only required change would be
to replace \( \mathbf{v}^e_i \) with velocity vectors pointing to pre-defined escape areas, which could be implemented dynamically in the simulation. However, any other crowd simulator can be used since it can adopt the velocity field generated in this work, as will be presented next.

### Using Tracked Trajectories to Obtain Desired Velocities

A fundamental problem for animating or simulating virtual humans during normal life is how to obtain desired velocities \( \mathbf{v}^e_i \) (intentions) for each agent \( i \), in order to mimic virtually situations observed in real life. In fact, a few approaches have been proposed in the past years, as discussed in the related work. However, these techniques are somewhat empirical, and require considerable interactivity. In this work, we use trajectories of tracked people in filmed video sequences as examples to feed our simulator in normal life situations, so that virtual agents tend to mimic the behavior of tracked people.

It is important to notice that our goal is to extract the actual intention of each tracked person. In denser crowds, the movement of each person includes not only their desired paths, but also a great amount of interactions with the environment and other people (in the sense of collision avoidance), as noticed in [28]. Consequently, the focus of this work is to use real life scenarios where people intentions can be better captured (i.e. non-dense crowds). Furthermore, we can increase the number of simulated people and include, in the simulation model, the expected interactions among people in the dense crowd using Equation (3). On the other hand, if we have a footage of a dense crowd (which contains
the inherent interactions among people), it is not easy to simulate the same scenario with a smaller number of agents without “undoing” the registered interactions.

Once we have the individual trajectories, the next step is to generate a velocity field that can provide the desired velocity for virtual agents at each position. There are different approaches to obtain dense vector fields from sparse ones, varying from traditional interpolation/extrapolation techniques, such as nearest neighbor, linear, cubic and splines [29] to more sophisticated methods, such as gradient vector fields [30]. In most of our experiments, there were no tracked trajectories at image borders, indicating that extrapolation techniques should be used. In fact, interpolation by nearest neighbor can be easily extended for extrapolation, and does not propagate the error in the extrapolated regions as much as other interpolation techniques (such as linear or cubic), because it is basically a piecewise-constant function.

In several scenarios, there are people walking in opposite directions (for instance, there are also some people moving downwards in the pathway illustrated in Figure 1). A single velocity field generated from all tracked trajectories would clearly lead to incoherent results, since the same pixel may present two velocity vectors pointing to opposite directions. Furthermore, the velocity field would probably contain several adjacent vectors with opposite directions, which would produce unrealistic oscillating trajectories. To solve such problems, we generate different “layers” of velocity fields, where each layer is produced using only coherent trajectories.

The definition of coherent (or similar) trajectories is very application-dependent. For
instance, Junejo et al. [31] used envelope boundaries, velocity and curvature information as features to group similar trajectories. Makris and Ellis [32] also used envelope boundaries to determine main routes from filmed video sequences. In both approaches, the spatial distance between trajectories is an important feature in the clustering procedure. For our purposes, coherent trajectories are those having approximately the same displacement vector (e.g. two trajectories going from left to right are coherent, regardless of their mean speed and the distance between them). For automatic classification and grouping of coherent trajectories, we first extract relevant features and then apply an unsupervised clustering technique, as explained next.

Let \((x(s), y(s)), s \in [0, 1]\), denote a trajectory reparametrized by arclength (normalized to unity), so that \((x(0), y(0))\) is the start point and \((x(1), y(1))\) is the end point of the trajectory. Each trajectory is then characterized by a set of \(N\) displacement vectors \(d_i = (\Delta x_i, \Delta y_i)\) computed at equidistant arclengths:

\[
d_i = (x(t_{i+1}) - x(t_i), y(t_{i+1}) - y(t_i)),
\]

where \(t_i = i/N\), for \(i = 0, \cdots, N - 1\). Then, each trajectory \(j\) is represented by a \(2N\)-dimensional feature vector \(f_j\), obtained by combining the \(N\) displacement vectors associated with the trajectory:

\[
f_j = (\Delta x_0, \Delta y_0, \Delta x_1, \Delta y_1, \cdots, \Delta x_{N-1}, \Delta y_{N-1}).
\]

Coherent trajectories tend to produce similar feature vectors \(f\). Hence, a set of coherent trajectories is expected to produce a cluster in the \(2N\)-dimensional space, which is mod-
eled as a Gaussian probability distribution characterized by its mean vector and covariance matrix. Since each cluster relates to a different Gaussian function, the overall distribution considering all feature vectors $f_j$ is a mixture of Gaussians. The number of Gaussians in the mixture (which corresponds to the number of clusters), as well as the distribution parameters of each individual distribution can be obtained automatically using the unsupervised clustering algorithm described in [33].

The number $N$ of displacement vectors used to assemble $f_j$ is chosen based on how structured the flow of people is. For relatively simple trajectories, small values of $N$ can capture the essence of the trajectories. On the other hand, more complicated trajectories (with many turns) are better characterized using larger values of $N$. In general, public spaces (such as sidewalks, parks, center towns, among others) tend to present main flow directions, and $N = 1$ or $N = 2$ are usually enough to identify different clusters. It should be noticed that, as $N$ increases, the dimension of the feature vectors increase, and a larger number of samples is needed for a reliable estimation of the Gaussian distributions. At the same time, unstructured motion (e.g. football players in a match, or kids playing), require a larger values of $N$ to summarize the desired trajectories, and a very large number of samples would be needed for the clustering algorithm. Hence, the proposed method is, in general, not suited for unstructured motion.

Figure 2(a) illustrates the clustering procedure applied to the trajectories illustrated in Figure 1(b). Trajectories going upwards and downwards were correctly placed into two different clusters, and were marked in different colors (red and green, respectively). The
corresponding extrapolated velocity fields are shown in Figures 2(b) and 2(c).

Finally, the desired velocity $v^i_g$ of agent $i$ is obtained from one layer of the extrapolated velocity fields $v^{\text{tracked}}$ at the agent’s current position, and Equation (3) is used to perform the simulation. It should be noticed that, when a virtual agent “is born”, it is associated with only one layer of velocity field. However, it can “see” all agents in all layers (and also information about the virtual environment such as walls and obstacles, which are accessible for all agents in all layers). Moreover, the radial repulsion force introduced by agents and obstacles in Helbing’s model (see Equation (3)) guarantees collision-free trajectories. Hence, agents tend to follow main directions provided by $v^{\text{tracked}}$, and at the same time they avoid collisions with obstacles and other agents. In fact, that is exactly our claim: for a few agents, they will follow mainly the extrapolated velocity fields (desired velocities); for denser simulated crowds, agents will have more interactions with other agents and obstacles, so that they probably will not be able to follow their desired trajectories.

It should be noticed that scenarios where people move with varying velocities can be successfully simulated by our algorithm. For instance, if a video sequence contains people walking with a different speed depending on the terrain (e.g. faster on dry surfaces and slower on wet/snowy terrains), the velocity fields would keep such characteristic, and simulated agents would try to reproduce approximately the same movements. In particular, the virtual reproduction of such kind of behavior requires a considerable amount of user interaction in competitive approaches (such as [17]), but can be done automatically in the proposed method. Results in the next Section illustrate this kind of behavior.
Results and Discussion

In this Section, we show some results to illustrate the potential of the proposed model. In particular, we show some quantitative comparisons between the real filmed sequences and the simulated version using the proposed model, and the transition from “normal life” to a panic situation.

The T-intersection Scenario

In this case study, we used a T-shaped region of our campus (T-intersection scenario). This scenario presents some diversity of trajectories, resulting in four trajectory clusters (layers), as illustrated with different colors in Figure 3 (circles indicate starting points of each trajectory).

This scenario presents particular characteristics that make people move with different speeds depending on the region they are walking on. In particular, there is a set of stairs (marked as region B in Figure 4(a)) that connects two roughly flat plateaus (for camera calibration and simulation purposes we considered that the whole region is planar, since the difference of the plane heights is small if compared to the height of the camera). In fact, Table 1 indicates that filmed people clearly slow down when climbing up or down the stairs when compared to flat regions (A and C in Figure 4(a)).

We simulated this environment with approximately the same number of people as in the filmed video sequence (a snapshot is depicted in Figure 4(b)). As shown in Table 1, the
simulated experiment presented results similar to the filmed sequence, and virtual humans also kept the tendency of reducing their speed in the region of the stairs. It should be emphasized that velocity fields used in this simulation were obtained in a fully automatic way (from people tracking to automatic clustering).

We also analyzed the influence of increasing the number of simulated agents, using the velocity fields obtained with a smaller number of people. Such experiments can be useful to predict the flow of people in public spaces during special occasions (e.g. a shopping mall near Christmas). We extrapolated the number of people for the T-intersection scenario from 20 to 150 agents, as illustrated in Figure 4(c). Average speeds in regions A, B and C for this experiment are shown in Table 1. As expected, such average speeds were reduced when compared to the experiment performed with 20 agents, particularly in region B. In fact, this crowded scenario produces traffic jams, which can be observed more evidently in regions with spatial constraints, such as the relatively narrow stairway. It is important to observe in the attached short movies that all crowds in this scenario (filmed, simulated with same number of agents as filmed sequence and extrapolating number of agents) present similar characteristics (spatial formation, decreasing velocities as a function of stairs and corridors).

The Downtown Scenario

This scenario represents a street in our city, where people move mainly from left-to-right and right-to-left, but they also get in and out of stores. Figure 5(a) shows the seven clusters
obtained with tracked trajectories, and Figure 5(b) shows a snapshot of the simulation.

Some quantitative measures in the observed (filmed) scenario and the simulated environment with approximately the same number of virtual humans (120 agents) were also compared for the downtown scenario. We measured the mean speed and standard deviation for people/agents that moved from the leftmost to the rightmost portion to the region (or vice-versa), illustrated by clusters in red and dark blue, respectively, in Figure 5(b). Such metrics are shown in Table 2, and they indicate that filmed and simulated scenarios present similar mean speeds.

**Transition from Normal Life to Panic**

As stated previously, the proposed model allows a very easy transition from “normal life” to a panic situation. During normal life, agents should follow the extrapolated velocity field obtained with computer vision algorithms. On the other hand, in panic situations, agents should rush to pre-defined exits. In fact, a very simple way to implement the transition model is to define the goal velocities by:

\[
\mathbf{v}_i^g = w_i(t)\mathbf{v}_i^{exit} + (1 - w_i(t))\mathbf{v}_i^{tracked},
\]

where \(\mathbf{v}_i^{tracked}\) is the velocity obtained from the extrapolated velocity field, \(\mathbf{v}_i^{exit}\) are pre-defined velocity vectors pointing to the closest exit to agent \(i\), and \(w_i(t)\) is a time-dependent weight function for agent \(i\). Such function should be zero during normal life (so that agents
would follow mostly the extrapolated vector field) or one when the agent perceives the panic situations (so that agents would run to interest points if a hazardous event is detected).

We simulated an emergency situation in the $T$-intersection scenario, by generating a panic situation after a few seconds of normal life (when parameter $w(t)$ is set to one). At this moment, all agents perceive the hazardous event (global alarm), and start running to pre-defined exits of environment in order to save their lives. Figure 6(a) shows a snapshot of normal life just before the emergency situation, while Figure 6(b) illustrates the situation 2 seconds after the panic situation started. (exits are marked with red ellipses).

**Conclusions and Future Work**

In this work, a new model for simulating virtual humans in normal life situations based on filmed video sequences was presented. Our main contribution was the introduction of computer vision algorithms to feed the crowd simulator with information obtained from actual video sequences. As far as we know, this is the first work that uses computer vision algorithms to improve the realism of the simulation in normal life situations. Furthermore, the proposed technique is fully automatic, opposed to other methods for simulating normal life situations of groups and crowds that require user intervention, such as [16–18].

We believe that we can use filmed video sequences of non-crowded structured environments to capture the essential intentions of each person, and then simulate such environments with the same number of agents (or a larger number) with the same intentions (i.e.
goal velocities). As the number of simulated agents increase, interaction forces generated by other agents, walls and obstacles (according to Equation (3)) become more intense, so that simulated agents must adapt their instantaneous velocities more strongly. It is important to notice that our algorithm was designed to simulate structured environments, where people tend to follow main directions (e.g. shopping malls, pathways, etc.), and it is not suited for non-structured movements (e.g. football players in a match). We also believe that it is more adequate to compute velocity fields using non-crowded video sequences, because in crowded scenes people may not be able to follow their desired paths due to interaction with other people (furthermore, the tracking procedure may present erroneous results in dense crowds).

Experimental results indicated that the proposed model can be effectively used to simulate virtual humans in normal life situations, keeping the same characteristics as the respective filmed video sequences. With our model, it is possible to increase the number of virtual agents in comparison with the filmed video sequence, as well to shift from normal life to a panic situation. As future work, we intend to investigate metrics for crowd comparisons, and extend our algorithms to simulate unstructured environments. Also, we intend to test our model by integrating it with other algorithms that handle interactions among people, like Reynolds [2] or Goldenstein [5], for instance.
Acknowledgements

This work was developed in collaboration with HP Brazil R&D.

References


<table>
<thead>
<tr>
<th>Region</th>
<th>Dir.</th>
<th>Video (speed) mean</th>
<th>std</th>
<th>Simulation, 20 (speed) mean</th>
<th>std</th>
<th>Simulation, 150 (speed) mean</th>
<th>std</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>→</td>
<td>0.96m/s</td>
<td>0.17m/s</td>
<td>0.84m/s</td>
<td>0.26m/s</td>
<td>0.81m/s</td>
<td>0.26m/s</td>
</tr>
<tr>
<td></td>
<td>←</td>
<td>1.00m/s</td>
<td>0.19m/s</td>
<td>0.91m/s</td>
<td>0.21m/s</td>
<td>0.90m/s</td>
<td>0.28m/s</td>
</tr>
<tr>
<td>B</td>
<td>↓</td>
<td>0.52m/s</td>
<td>0.33m/s</td>
<td>0.48m/s</td>
<td>0.30m/s</td>
<td>0.35m/s</td>
<td>0.31m/s</td>
</tr>
<tr>
<td></td>
<td>↑</td>
<td>0.53m/s</td>
<td>0.29m/s</td>
<td>0.58m/s</td>
<td>0.29m/s</td>
<td>0.31m/s</td>
<td>0.27m/s</td>
</tr>
<tr>
<td>C</td>
<td>→</td>
<td>1.03m/s</td>
<td>0.20m/s</td>
<td>0.89m/s</td>
<td>0.27m/s</td>
<td>0.78m/s</td>
<td>0.33m/s</td>
</tr>
<tr>
<td></td>
<td>←</td>
<td>1.06m/s</td>
<td>0.20m/s</td>
<td>0.99m/s</td>
<td>0.23m/s</td>
<td>0.85m/s</td>
<td>0.34m/s</td>
</tr>
</tbody>
</table>

Table 1: Quantitative metrics for evaluating the *T-intersection* scenario with 20 and 150 agents.

Figure 1: (a) Initial detection of a person, and determination of the correlation template. (b) Several tracked trajectories (circles indicate start-points).
Table 2: Mean speed and standard deviation to cover the horizontal space for the *downtown* scenario, for filmed sequence and simulated environment

<table>
<thead>
<tr>
<th>Direction</th>
<th>Mean speed</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filmed</td>
<td>→</td>
<td>1.33m/s</td>
</tr>
<tr>
<td>Simulation</td>
<td>→</td>
<td>1.33m/s</td>
</tr>
<tr>
<td>Filmed</td>
<td>←</td>
<td>1.13m/s</td>
</tr>
<tr>
<td>Simulation</td>
<td>←</td>
<td>1.15m/s</td>
</tr>
</tbody>
</table>

Figure 2: (a) Result of trajectory clustering applied to Figure 1(b). (b)-(c) Extrapolated velocity fields for the two trajectory clusters showed in (a)
Figure 3: Different layers of velocity fields for the *T-intersection* scenario.

Figure 4: (a) Regions used for quantitative validation. (b) Snapshot of simulation with a small number of agents. (c) Snapshot of simulation with a large number of agents.
Figure 5: (a) Trajectory clusters obtained for the *downtown* scenario. (b) A snapshot of the simulation using the clusters shown in (a).
Figure 6: (a) Normal life behavior for the \textit{T-intersection} scenario. (b) Panic situation behavior for the \textit{T-intersection} scenario.