Fast-Forwarding Crowd Simulations

Cliceres Mack Dal Bianco[1], Adriana Braun[1], Soraia Raupp Musse[1⋆], Claudio Jung[2] and Norman Badler[3]

[1] Graduate Program in Computer Science
Pontifical Catholic University of Rio Grande do Sul - PUCRS - Porto Alegre, Brazil
[2] Graduate Program in Computer Science
Federal University of Rio Grande do Sul - UFRGS - Porto Alegre, Brazil
[3] University of Pennsylvania - UPENN -
Philadelphia - USA

Abstract. The processing time to simulate crowds for games or simulations is a real challenge. While the increasing power of processing capacity is a reality in the hardware industry, it also means that more agents, better rendering and most sophisticated Artificial Intelligence (AI) methods can be used, so again the computational time is an issue. Despite the processing cost, in many cases the most interesting period of time in a game or simulation is far from the beginning or in a specific known period, but it is still necessary to simulate the whole time (spending time and processing capacity) to achieve the desired period of time. It would be useful to fast forward the time in order to see a specific period of time where simulation result could be more meaningful for analysis. This paper presents a method to provide time travel in Crowd Simulation. Based on crowd features, we compute the expected variation in velocities and apply that for time travel in crowd simulation.

1 Introduction

Time travel is the concept of movement between certain points in time, analogous to movement between different points in space. While the theory of CTC (Closed Time-like Curves) is studied, it is not currently feasible for real human travel with current technology [5]. However, in computer science it may be possible to fast forward behaviors. That is exactly the goal of this paper: to fast forward the behaviors of crowds. The time travel in crowds can have many applications. In games, a nice example is the "fog of war" [7]. This term in video games [1] refers to enemy units, and often terrain, being hidden from the player. In Real Time Strategy (RTS) games, units may navigate in partially unknown and non-visible environments, searching for enemies. Imagine that non-visible enemies can have sophisticated behaviors, e.g. navigation, searching and escaping in realistic way. However, such enemies should only be visible in a specific time. The challenge in this case is the computational time, i.e. to compute realistically non-visible behaviors has a huge compromise with the processing time. In this paper, we

⋆ e-mail:soraia.musse@pucrs.br

are interested in fast forwarding the crowd simulation, while keeping the realism of behaviors that have not been computed. Another important application is security. Suppose that you know the situation of a crowd in time t but you want instantaneously to know about this crowd in time $t+\Delta t$. If the predicted behavior might lead to security risks, the safety staff could take action in time t to change the (potentially dangerous) results in time $t + \Delta t$. The challenge of this work is to try to preserve the crowd motion during the travel time, in order to have realistic results and minimize the error occurred due to the fast approximation of the actual simulation. Important aspects that can disturb an agent in the crowd to achieve its goal in the desired time has to be considered, such as collisions with other agents, obstacles and world complexity. In this work, we are focused on computing the disturbance that avoiding collisions among people caused in agents motion, and then use such information to generate plausible speeds to be applied during the time travel, where the simulation is turned off. Indeed, as the prior for the motion, we estimate agents position with Dead Reckoning techniques, which apply Physics to estimate positions, and then consider other agents and environment complexity. Dead Reckoning is a relatively new navigation technique [16, 2], that starting from a known position, generates successive positions displacements. Pedestrian Dead Reckoning (PDR) is the estimation of walking speed and a direction of walking. Although there is an extensive body of research on this subject [9, 13], the major part of such methods are focused on pedestrian positioning in real life. Furthermore, as far as we know, none of these methods aimed to fast forward time in crowd simulation. This is the novel idea of this work.

2 Related Work

Some methods in crowd simulation have been presented in the literature aiming to reduce the complexity of crowd collision avoidance. Guy et al. [6] extends the notion of velocity obstacles from robotics for collision free navigation in a parallelized algorithm. Pettre and collaborators [11, 12] propose some methods to improve the performance and time of collision avoidance processing. Osborne [10] introduces a hierarchical approach for presenting agent behavior details according to the camera's distance focused on 3D animation of agent. Bianco [4] describes a method to fully eliminate the crowd collision and preserve the motion of crowd during a specific period of time. Although these works discuss the effect of reducing the complexity or even eliminate the computational time of crowd collision, they are not focused on travel time and motion preservation in crowds, as the main goal of our work.

The position of a person is a valuable information used in many applications ranging from vehicle navigation, ambient assisted living, location-dependent advertisement, among many others [16]. Although viable techniques exist in indoor and outdoor environments, the accuracy of devices and methods is still a challenge [2]. The use of dead-reckoning in computer graphics is not novel. In 1997 [15], the authors propose a way to decrease the amount of messages commu-

nicated among the participants in a multiplayer simulation system. Hakiri and collaborators [8] propose ANFIS Dead Reckoning, which stands for Adaptive-Network-based Fuzzy Inference Systems Dead Reckoning. The proposed mechanism is based on the optimization approach to calculate the error threshold violation in networking games. Our work aims to use a dead reckoning system to estimate agents' positions in a future time, in the context of crowds. In addition, we want to provide instantaneous estimations of the positions, i.e. without compromising the computational time. As far as we know this has not been used in the context of crowds simulation.

3 The Model for Time Machine

In this section we present our approach to provide a time machine for crowds. Our method has a training step which consists of learning how each agent is affected by the others. We called this as IP step (line 7 in Algorithm 1), i.e. how individual velocities should be affected by the presence of others. In addition to that term, crowd position estimation should also take into account the environment complexity (EC - line 6), i.e. the free region and presence of obstacles. PDR (line 5) represents the dead reckoning method using physics to estimate positions for agents in the crowd, in the future. These three methods work sequentially to obtain a estimated position for each agent in the crowd. Finally, the last step *Repositioning* (line 11) is responsible for locating the agents in the environment, avoiding collisions among them and obstacles. The overall algorithm of the method is following specified: Our model of time machine can be

Algorithm 1 Time Machine

1: **procedure** TM
2: Continuous Simulation stops at frame t;
3: Extract data from environment and the position of agent i in frame t, called pos_t^i;
4: *loop*: For each agent i at frame $t + \Delta t$
5: $pos^i_{t+\Delta t}(x^i_{t+\Delta t}, y^i_{t+\Delta t}, z^i_{t+\Delta t}) \leftarrow PDR(x^i_t, y^i_t, z^i_t)$;
6: $pos^i_{t+\Delta t}(x^i_{t+\Delta t}, y^i_{t+\Delta t}, z^i_{t+\Delta t}) \leftarrow EC(x^i_{t+\Delta t}, y^i_{t+\Delta t}, z^i_{t+\Delta t})$;
7: $pos^i_{t+\Delta t}(x^i_{t+\Delta t}, y^i_{t+\Delta t}, z^i_{t+\Delta t}) \leftarrow IP(x^i_{t+\Delta t}, y^i_{t+\Delta t}, z^i_{t+\Delta t})$;
8: $i \leftarrow i + 1$.
9: **goto** *loop*.
10: **close**;
11: Repositioning($\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$);
12: End.

integrated with any crowd simulator, since the required data can be provided in time t. However, there is an error associated with the estimation process in our method. The main reason is that our prior is computed based on PDR, which can accumulate errors. PDR considers constant speed, but in crowds desired speeds usually cannot be achieved because of the interaction with obstacles and other agents. In order to test our model, we used BioCrowds [3]. It was chosen because it is a free of collision algorithm and there is an available implementation. In

Section 3.1 we present some details about the software. In addition, the next sections describe details of all steps.

3.1 BioCrowds

The method for crowd simulation called BioCrowds proposed by Bicho et al [3] is based on the space colonization algorithm presented by Runions et. al [14]. This last algorithm models leaf venation patterns and branching architecture of trees, and it operates by simulating the competition for space guided by the distribution of markers that mimic the auxin hormone. The BioCrowds simulator uses the same idea in order to simulate agents' paths in crowds: the distribution of markers guide agents' paths by indicating the free space and possibility of movement. The simulation environment is populated with discrete markers over "walkable" regions and, for each agent i, individual parameters are assigned, namely its current position $\boldsymbol{X}_t^i$ its current goal $\boldsymbol{g}^i$, its desired maximum speed s_{max}^i and its perception field (the maximum distance from which an agent can perceive markers), modeled as a circular region with radius R_i. At each simulation step the position $\boldsymbol{X}$ and goal objective $\boldsymbol{g}$ pointing to the agent's goal are update synchronously. Thus, the algorithm can be divided into two steps: i) computation of nearest markers from each agent and ii) computation of motion direction for each agent. Agents compete for fixed markers, allocating and freeing them as they move throughout the space, allowing other agents to occupy their previous space. It results in a free-of-collision method to simulate crowds. In the case of this paper, the characteristic of BioCrowds to adjust the desired speed based on available space can increase the error in PDR method, since variations in speed bring the estimation far from the constant speed one. However, since it is a common effect in crowd simulations (and in real life), we decided to keep BioCrowds in our tests.

3.2 Dead Reckoning in the Crowd ($\boldsymbol{PDR}$)

The method used for PDR is based on physics, considering a previously determined position, direction to the goals and constant speed. Dead reckoning can give the best available information on position, but is subject to significant errors due to many factors such as varying speed, obstacles and other agents. In order to estimate a future position for agent i in frame $t+\Delta t$, our PDR method needs: *i)* the position of agent i in frame t, denoted by $\boldsymbol{X}_t^i = (x_t^i, y_t^i, z_t^i)$; *ii)* the agent goal $\boldsymbol{g}^i = (g_x^i, g_y^i, g_z^i)$; and *iii)* its desired speed s^i. Given the required parameters, the estimated position of agent t in frame $t + \Delta t$ is given by

$$\boldsymbol{pdr}_{t+\Delta t}^i = \boldsymbol{X}_t^i + \left(s^i \frac{\boldsymbol{g}^i}{||\boldsymbol{g}^i||} \right) (\Delta t), \tag{1}$$

which is a linear estimate using constant velocity assumption. In order to measure the error caused by PDR, we firstly did a simulation with only one agent. Input simulation data is presented in Table 1. It is important to mention that

the goal does not represent the desired position at frame $t + \Delta t = 700$, in fact, the travel time can be to any frame in the future, until the agent achieves the goal. Based on such data, we simulated one agent in an environment without obstacles, and measured the error presented in the PDR estimation in comparison to BioCrowds. More precisely, we computed the relative error given by

$$Error^i_{t \to t+\Delta t} = \frac{d(\boldsymbol{X}^i_{t+\Delta t}, \boldsymbol{pdr}^i_{t+\Delta t})}{d(\boldsymbol{X}^i_t, \boldsymbol{X}^i_{t+\Delta t})}, \tag{2}$$

where $\boldsymbol{X}^i_t$ is the position of agent i in frame t (when simulation stops and the time machine starts), while $\boldsymbol{pdr}^i_{t+\Delta t}$ and $\boldsymbol{X}^i_{t+\Delta t}$ are the positions of the same agent in frame $t+\Delta t$ using the time machine and the full continuous simulation, respectively. For the tested simulation the achieved error was 0.0013.

Table 1. Simulated data with only one agent.

Frame t	100
Position at frame t	$(5.453, 20.181, 0.0)$
Speed s	0.046/frame
Goal $\boldsymbol{g}$	$(38.039, 19.979, 0.0)$
Frame $t + \Delta t$	700

3.3 Computing the Environment Complexity (EC)

Although it is important to start with a good prior, in this case the PDR estimation, it is also relevant to consider the complexity of the environment in a time machine model. Clearly, the free navigation area, as well as the number and area of obstacles, have impact on the flow of pedestrians. The environment complexity is modeled taken into account the free space for crowd movement, i.e. considering number of agents and obstacles and the space needed to place all of them. We included a constant $0 < \alpha < 1$ that describes the weight to be given to the number of obstacles, if one wants to consider that in addition to the sum of obstacles areas.

$$EC = \min\{1, \frac{n_a A_a + \alpha n_o}{A_w - A_o}\}, \tag{3}$$

where n_a represents the total of agents in the simulation; A_a is the area of one agent and n_o is the number of obstacles in the world. Also, A_w represents the area of the world and A_o is the sum of all obstacles areas. If the number of agents or area occupied by obstacles are too big and there is no free space to allow individuals motion, then EC is truncated at 1 and no motion is allowed. Equation 4 presents our proposal to use EC as a penalty function in the generation of each agent's position in the time machine model when PDR and the complexity of environment EC are considered, obtaining the final PDR estimation:

$$\boldsymbol{pdr'}^i_{t+\Delta t} = \boldsymbol{pdr}^i_{t+\Delta t} - \left(s^i \frac{\boldsymbol{g}^i}{||\boldsymbol{g}^i||} \right) EC \Delta t \tag{4}$$

$$= \boldsymbol{X}_t^i + \left(s^i \frac{\boldsymbol{g}^i}{||\boldsymbol{g}^i||} \right) (1 - EC) \Delta t.$$

3.4 Computing the Interaction among People in Crowds (IP)

In crowd simulation, the interaction among agents due to the intrinsic collision avoidance processes can produce a disturbance in the desired velocity of each agent i. This disturbance depends on various factors such as the direction of the flows, desired speed variability and local density, among others. Due to the complexity in the simulation of dense crowds, we decided to adopt a stochastic approach to learn the disturbance arisen from the interaction of the agents. In this approach, we observed the impact of the local density and of counter-flow in the effective velocity of the agents.

To this end, we performed several pairs of simulations, varying the number of agents n_a and the average and standard deviation of the desired speed in the scenario of Figure 3. Each pair of simulations corresponds to one simulation with n_a agents moving in an uni-directional flow and another with $2n_a$ agents moving in a bi-directional flow. The first n_a agents from the bi-directional simulation have the same id, starting point and desired velocity than the agents of the uni-directional flow. In order to measure the disturbance caused by the counter-flow in the movement of a given agent i, we computed the displacement of this agent in a time-window $W = 2$s in both simulations of each pair uni/bi-directional. Let us call the displacement of agent i in the uni-directional flow $\Delta \boldsymbol{X}_{iU}$, and the displacement of the equivalent agent in the bi-directions flow $\Delta \boldsymbol{X}_{iB}$. The impact in the speed of agent i from the disturbance caused by the counter-flow is given by:

$$\Delta \boldsymbol{v}_i = \frac{\Delta \boldsymbol{X}_{iU} - \Delta \boldsymbol{X}_{iB}}{W}. \tag{5}$$

We measured the values of $\Delta \boldsymbol{v}_i$ for each agent in the different pairs of simulations. For each one of them we associated the corresponding value of local density of agents, computed considering the number of agents present in a 4m^2 square region, with the agent i at the center. We grouped the values of $\Delta \boldsymbol{v}_i$ as a function of the values of local densities $\rho \in \{0.25, 0.50, ..., 2.5\}\text{agents/m}^2$. The distribution of speed variation from all pairs of BioCrowds simulations for $\rho = 1\text{agent/m}^2$ is shown by the dark columns of the histogram in Figure 1. Then, for each value of ρ, we analyzed the frequency distribution of $\Delta \boldsymbol{v}_i$. We found that those distributions resemble the Weibull distribution, given by:

$$f(x) = \frac{b}{a} \left(\frac{x}{a} \right)^{b-1} \exp \left[- \left(\frac{x}{a} \right)^b \right] \tag{6}$$

We used maximum likelihood estimation in order to find the values of a and b that best fit the Weibull distribution to the data. After this process, we achieved 10 Weibull $f(\Delta \boldsymbol{v}|\rho)$ distributions, one for each value of local density ρ. Once we get the distributions of speed reduction, we can generate random values r_i for speed reduction, according to the local density of agent i. This reduction is

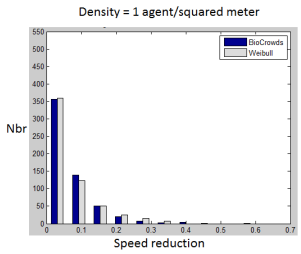

Fig. 1. Histograms of speed reductions in pairs of BioCrowds simulations and randomly generated speed reductions according to Weibull distributions (in m/s).

incorporated to adjusted PDR estimates, and the position $\boldsymbol{tm}^i_{t+\Delta t}$ produced by the time machine for each agent i is then given by

$$\boldsymbol{tm}^i_{t+\Delta t} = \boldsymbol{pdr'}^i_{t+\Delta t} - r_i \frac{\boldsymbol{g}^i}{||\boldsymbol{g}^i||} \Delta t. \tag{7}$$

3.5 Re-positioning the Agents in Crowds

Our model for the time machine does not need any simulated trajectory to be fast forwarded. Indeed, we propose penalty functions and generate positions in a future time only based on information available when simulation stops (time t used in this text). This fact consumes least processing time than if a path planning algorithm must be executed for each agent in the simulation. However, the drawback of avoiding the use of simulated information in the time machine is that the position $\boldsymbol{tm}^i_{t+\Delta t}$ generated for agent i in future time $t+\Delta t$ can be placed in an impossible location, such as the interior of an obstacle or overlapping with another agent that had been already placed.

In order to treat this problem, we implement an extension to original BioCrowds to read the positions $\boldsymbol{tm}$ of agents and consider those as desired positions, and not final ones. In fact, BioCrowds tries to achieve the desired positions, but if it is impossible, it tries to place them in the nearest possible region. Due to the model inherent structure to compete for markers (see Section 3.1), the agents are placed exactly in the position $\boldsymbol{tm}$ when this position has free markers to be allocated. Otherwise, the algorithm searches for free markers in the four cardinal directions (N, E, S, W) oriented w.r.t. to the goal vector and shifted by R_i (perception field for agent i, i.e. the maximum distance from which an agent can perceive markers), as explained in Section 3.1.

Clearly, this procedure impacts in errors that are evaluated in the next section, but we consider that as acceptable in order to provide the time machine instantaneously and without the needed to simulate any trajectory in the travel time between frames t and $t + \Delta t$.

4 Experimental Results

In order to evaluate our model we tested the time machine in one environment with 920sqm (23×40meters) and four obstacles configurations: *i)* without obstacles; *ii)* with 4 obstacles and $A_o = 582.22$sqm; *iii)* with 2 obstacles and $A_o = 109.71$sqm; *i)* with 6 obstacles and $A_o = 128.68$sqm. In addition, these four scenarios were simulated with populations of $8, 80, 160, 240$ and 320 agents, and each agent has a diameter of 0.456m. For all scenarios, we computed the relative error in position for each agent i according to Equation (8) (similar to Equation (2), except that $\boldsymbol{tm}^i_{t+\Delta t}$ is used instead $\boldsymbol{pdr}^i_{t+\Delta t}$). The average distances for each scenario for all agents in each scenario are presented in Figure 2.

$$Dif^i_{t \to t+\Delta t} = \frac{d(\boldsymbol{X}^i_{t+\Delta t}, \boldsymbol{tm}^i_{t+\Delta t})}{d(\boldsymbol{X}^i_t, \boldsymbol{X}^i_{t+\Delta t})}, \tag{8}$$

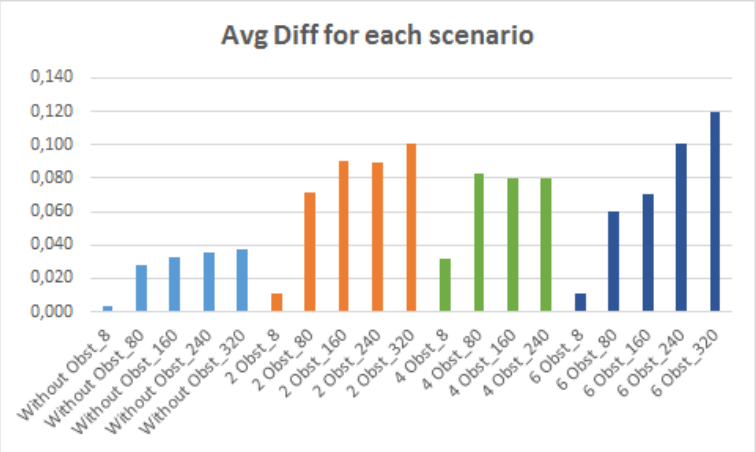

Fig. 2. Average distance (meters) from all agents for each scenario, when comparing the position generated in our model and the continuous simulation.

As can be seen in Figure 2, the maximum achieved error is approximately 0.12 in a scenario with 320 agents and 6 obstacles. In this scenario the average error value is 0.06. To assess the visual quality of our results, the following images illustrate simulation results. The Figure 3 shows 3 images of a simulated scenario without obstacles, containing 240 agents. On the left, the image represents the frame the simulation stops (frame 350) and arrows indicate the motion direction of groups. In the center we show frame 650 with the continuous simulation and the predicted result using the proposed time machine (right). As can be observed, there is a visual difference. It can be explained due to PDR and also due to the fact that time machine positions are not free of collision, and then BioCrowds

should change positions to keep agents away from the others, as described in Section 3.5.

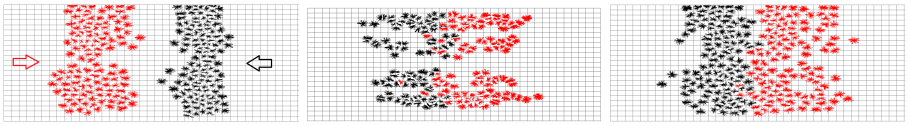

Fig. 3. Left: Frame the simulation stops (frame 350). Center: frame 650 using the continuous simulation and the proposed time machine (right).

In the next example, we considered the scenario with larger error, according to Figure 2, with 4 obstacles and 240 agents [4]. Figure 4 shows 3 images of the simulated scenario and 4 obstacles. On the left, the image represents the frame the simulation stops (frame 200) and arrows indicate the motion direction of groups. In the center, the image illustrates the frame 370 without time machine, while on the right the image is the frame 370 with time machine. This is the scenario with numerical error of 0.08 (Figure 2).

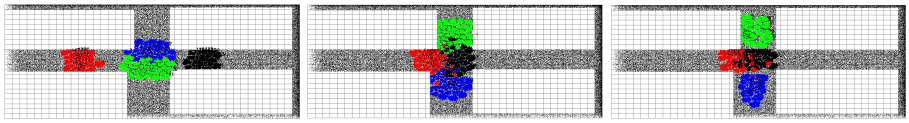

Fig. 4. On the left: the image represents the frame the simulation stops (frame 200), on the center: image illustrates the frame 370 without time machine, while on the right the image is the frame 370 with time machine.

5 Final Considerations

This paper presents an approach for fast forwarding crowd simulations. Our method considers the environment complexity in Pedestrian Dead Reckoning (PDR) and proposes a way to take into account the people interaction. The method was tested using BioCrowds, but can be adapted to any crowd simulation algorithm. There are points that should be investigated in further analyses: local EC instead of global EC, more agents and larger environments, more complex obstacles, other people distributions for disturbance and the integration with other crowd simulators.

References

1. Adams, E.: Fundamentals of Game Design. New Riders Press, Berkeley, CA (2014)
2. Beauregard, S., Haas, H.: Pedestrian dead reckoning: A basis for personal positioning. In: PROCEEDINGS OF THE 3rd WORKSHOP ON POSITIONING, NAVIGATION AND COMMUNICATION (WPNC06). p. 1 (2006)

[4] This scenario could not be simulated with 320 agents because the agents were stuck due to lack of free space

3. Bicho, A.L., Rodrigues, R.A., Musse, S.R., Jung, C.R., Paravisi, M., Magalhes, L.P.: Simulating crowds based on a space colonization algorithm. Computers and Graphics 36(2), 70 – 79 (2012), http://www.sciencedirect.com/science/article/pii/S0097849311001713, virtual Reality in Brazil 2011
4. Cliceres Mack Dal Bianco, Jovani Oliveira Brasil, A.B., Musse, S.R.: A model to compute people disturbance in crowds. In: Poster at ACM Motion in Games, 2015. p. 1 (2015)
5. Everett, A.: Time travel paradoxes, path integrals, and the many worlds interpretation of quantum mechanics. Phys. Rev. D 69, 124023 (Jun 2004), http://link.aps.org/doi/10.1103/PhysRevD.69.124023
6. Guy, S.J., Chhugani, J., Kim, C., Satish, N., Lin, M.C., Manocha, D., Dubey, P.: Clearpath: Highly parallel collision avoidance for multi-agent simulation. In: ACM SIGGRAPH/EUROGRAPHICS SYMPOSIUM ON COMPUTER ANIMATION. pp. 177–187. ACM (2009)
7. Hagelback, J., Johansson, S.: Dealing with fog of war in a real time strategy game environment. In: Computational Intelligence and Games, 2008. CIG '08. IEEE Symposium On. pp. 55–62 (Dec 2008)
8. Hakiri, A., Berthou, P., Gayraud, T.: Qos-enabled anfis dead reckoning algorithm for distributed interactive simulation. In: Turner, S.J., Roberts, D.J. (eds.) DS-RT. pp. 33–42. IEEE Computer Society (2010), http://dblp.uni-trier.de/db/conf/dsrt/dsrt2010.html#HakiriBG10
9. Ladetto, Q., Merminod, B.: Digital magnetic compass and gyroscope integration for pedestrian navigation. In: 9th International Conference on Integrated Navigation Systems, St-Petersburg. pp. 27–29 (2002)
10. Osborne, D., P., D.: Improving games ai performance using grouped hierarchical level of detai. Elsevier Science Inc., New Yor
11. Pettre, J., Ciechomski, P.d.H., Maim, J., Yersin, B., Laumond, J.P., Thalmann, D.: Real-time navigating crowds: scalable simulation and rendering. Computer Animation and Virtual Worlds 17(3-4), 445–455 (2006), http://dx.doi.org/10.1002/cav.147
12. Pettr, J., Ondrej, J., Olivier, A.H., Crtual, A., Donikian, S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In: Fellner, D.W., Spencer, S.N. (eds.) Symposium on Computer Animation. pp. 189–198. ACM (2009), http://dblp.uni-trier.de/db/conf/sca/sca2009.html#PettreOOCD09
13. Randell, C., Djiallis, C., Muller, H.: Personal position measurement using dead reckoning. In: In Proceedings of The Seventh International Symposium on Wearable Computers. pp. 166–173. Springer (2003)
14. Runions, A., Fuhrer, M., Lane, B., Federl, P., Rolland-Lagan, A.G., Prusinkiewicz, P.: Modeling and visualization of leaf venation patterns. In: ACM SIGGRAPH 2005 Papers. pp. 702–711. SIGGRAPH '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1186822.1073251
15. Tolga K. Capin, Igor Pandzic, N.M.T., Thalmann1, D.: A dead-reckoning algorithm for virtual human figures. Proceedings of VRAIS97 1(1), 161–169 (1997)
16. Zampella, F., Ruiz, A.R.J., Granja, F.S.: Indoor positioning using efficient map matching, rss measurements, and an improved motion model. IEEE Transactions on Vehicular Technology 64(4), 1304–1317 (April 2015)