Generating Background Population for Games based on Real Video Sequences

Marlon F. de Alcantara* Estevao S. Testa Gabriel L. B. da Silva Rodolfo M. Favaretto
Leandro L. Dihl Soraia Raupp Musse

Pontifical Catholic University of Rio Grande do Sul, Computing Science Department, Brazil

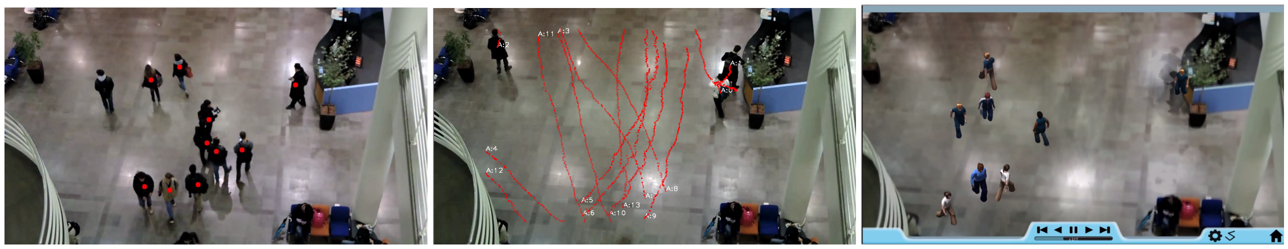
Figure 1: Overview of our method: on the left side real people trajectories are detected and tracked in a video sequence; in the center the trajectories are used to learn patterns of motion in our method, and on the right side our viewer illustrates population for a game, based on real life trajectories.

Abstract

There are several types of non-player characters in current games. They can be endowed with different levels of intelligence and their action/motion can present more or less realism. In this scope, it is still a challenge to create, in an automatic and easy way, NPCs that move and behave in a way that seems natural to the player. Known issues include repetitive and non-coherent trajectories for NPCs. In addition, background characters must be defined some way per designers, consuming time and resources. Facing this, we present a trajectory learning system derived from real video sequences to reproduce easy controlled NPCs for games. Our goal is to generate individual trajectories without any needed human intervention and based on real life population. The system is validated for a set of real environments, but can be scaled for any environment that have video sequences and tracked human trajectories for training. Results indicate that learning techniques can be useful to automatically generate NPCs trajectories.

Keywords: Background population, human trajectories, computer vision.

1 Introduction

Modern technologies have provided greater realism in computer games. However, most of the current game environments still need to better reflect the real world. In particular, most Non-Playable Characters (NPCs) are limited by a restricted range of trajectories, defined by the game designers. This aspect, relevant in background population, when unrealistic, can make the game boring for the players. Moreover, the background characters should be programmed in someway by game designers, that spend time and resources with this task, that can be tedious.

The main goal of this paper is to propose a way to generate individuals behaviors based on learning trajectories from real video sequences. It is not the first time computer vision and games are related. As presented by Freeman et al [6] in 1996, ”the appeal of computer games may be enhanced by vision-based user inputs”. Certainly it was the case of last decades with the usage of computer vision in interfaces.

Another example of computer vision in games is the work proposed by Fink and collaborators [5]. The authors describe an application to learn the behavior of NPCs (and other entities) in a game when observing just the graphical output of the game during game play. In this case, the authors used computer vision techniques to extract information from a played game.

As far as we know, the integration between computer vision and games for NPCs trajectories control is still incipient in the area. Section 2 discusses some related work. The main contribution of this paper is to use computer vision fo track and learn patterns from real video sequences. Such patterns can be used to generate individual trajectories for games, in an automatic way.

The remaining of this paper is organized as follows: Section 3 presents the method we adopt to detect and track people in video sequences and Section 4 describes our proposal to learn and estimate new trajectories for NPCs. Lastly, we developed an Unity application that allows to read a short video and provide a continuation for that, based on the learning trajectories. This application allows to visualize the whole film, partially from real life, partially computer-generated. Such aspects are discussed in Section 5.

2 Related Work

According to Moon et al. [14], the most Non-Playable Characters (NPCs) are limited by a restricted range of reactions within the playing parameters of the game, which can be boring for the

*e-mail: marlonmfa@gmail.com

players. As a result, the players may quickly lose their interest in the game. The methods for both effectively generating and controlling behavior are indispensable.

Currently, the NPCs are increasingly complex, this is because there is a need to make the games more believable, natural and human-like. Several approaches apply Artificial Intelligence (AI) techniques to control the behavior of the NPCs [1, 23, 9]. The authors in [9] proposed a genetic algorithm based on robotic controller to control one or more NPCs, in a dynamic way. Akbar et al. [1] employ a Fuzzy method for team coordination based on smart agent as a leader in attacking scenario. Gaussian distribution is used to make variation action of each NPC more natural and unpredictable.

However, according to Anderson [2], the major part of the artificially intelligent behavior displayed by game characters is hard-coded into the game program itself, and any needed change leads games programmers to spend a large amount of development time.

When the NPCs are not a change effector in the game, for example, a crowd walking in a determined local (background population), the use of a model based on AI is not necessary [11] and then, the approaches based on scripting are very suitable. However, by using scripted behaviors some reactions are hardly natural and human-like. Thus, we proposed to use computer vision techniques to extract people trajectories to be automatically used to control NPCs behaviors.

In fact, crowd simulation based on real video sequences are not new. When we generate individuals behaviors based on learning trajectories from real video sequences, detecting crowd features and the type of crowd existent in a given video sequences is usually known as data driven crowd simulation. Furthermore, detecting the type of a crowd in video sequence can be useful to help simulating coherent crowds and many methods have been proposed in this area [12, 13, 16, 8, 17, 10].

3 Method for Extracting Trajectories

The people initial detection is performed using the work proposed by Viola and Jones [21]. The boosted classifier working with Haar-like features was trained with 4500 views of people heads as positive examples, and 1000 negative (CoffeBreak and Caviar Head datasets from Tosato et al. [19] were used). This detector performs the initial position detection of people based on their heads, which are the input parameters for the next step: tracking. Once the individuals are detected, the trajectories are obtained using a method proposed by Bins et al. [3].

This approach for object tracking is based on multiple disjoint patches obtained from the target. The patches are represented parametrically by the mean vector and covariance matrix computed from a set of feature vectors that represent each pixel of the target. Each patch is tracked independently using the Bhattacharyya distance [7], and the displacement of the whole template is obtained using a Weighted Vector Median Filter (WVMF).

To smooth the trajectory and also cope with short-term total occlusions, a predicted displacement vector based on the motion of the target in the previous frames is also used. The appearance changes of the target are handled by an updating scheme. The output of tracking is a vector of each person i with the positions $\vec{X}_i^f = (x_i, y_i)$, at each frame f. It should be noticed that tracking is not a contribution of this work, and any pedestrian/crowd tracking algorithm can be used in this phase. Figure 2 illustrates the output of this step.

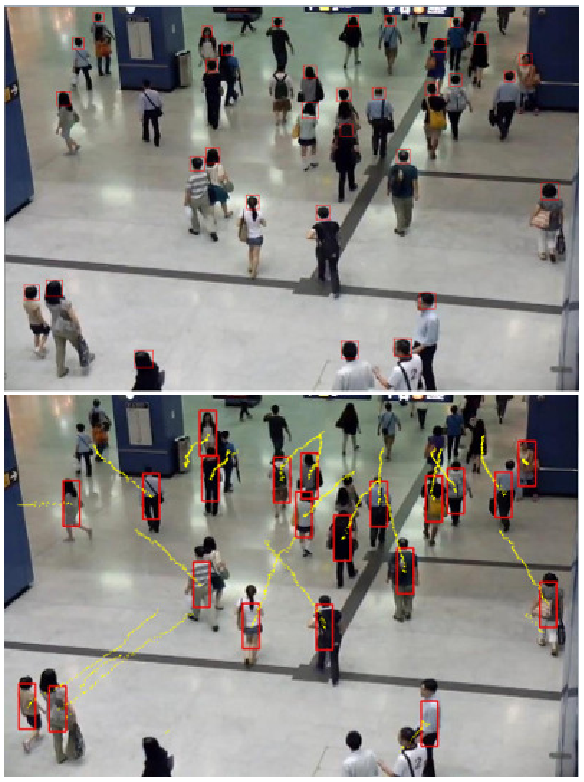
Figure 2: Tracking phase in our approach. On the top: the input image and heads detection, and on the bottom: the people detection and tracked trajectories.

4 Method for Learning and Simulating Trajectories

Currently, sophisticated graphics interfaces are available on commercial softwares (e.g. Autodesk) in order to control characters trajectories. Basic typical parameters are the initial locations and desired goals, distributed in some way among the virtual agents.

In the case of this work, we statistically distribute the agents according to the learned patterns from real life video sequences. These factors can directly influence the resulting trajectories the agents will apply. In order to improve the realism of trajectories in games, we propose to use extracted data from real life. Based on tracked trajectories, we learn the motion patterns and provide ways to reproduce such trajectories in a virtual environment.

4.1 Analyzing Real Trajectories

As stated in Section 3, for each individual i, at each frame f in the video sequence, we extracted $\vec{X}_i^f = (x_i, y_i)$. In addition, i is classified during the video sequence regarding its presence in the video ending. There are two possible classifications. The first one (class A) happens when individual i is present in the video ending, in this case his/her individual trajectory can be continued, i.e. an agent is virtually created and should keep going on the individual path. The idea behind has the goal to provide a continuity between the real and virtual sequences.

The second possible class (B) is defined for individuals which trajectories do not need to be continued. In this case, their trajectories are used to guide the agents in the virtual world.

Let us consider an individual trajectory as a set of positions $\vec{X}_i$ in 2d world with F elements, representing the F frames of motion duration. The set of trajectories in a specific video sequence is used in order to learn the existent patterns. In fact, the individuals trajectories extracted from video sequences are used to defined the set of possible movements the agents will apply in the virtual environment.

For a specific agent a, associated to an individual i in the video, the set of of possible moves (MOV_a) considering the trajectory applied by i is given by Equation 1:

$$MOV_a = \bigcup_{f=1}^{F-1} \{\Delta\vec{X}_i^f\}, \tag{1}$$

where $\Delta\vec{X}_i^f = \vec{X}_i^{f+1} - \vec{X}_i^f$. Next sections detail the process of creation and simulation of agents.

4.2 Creating Virtual Agents

Firstly, we define the initial agent a attributes as follows:

- Identity (id_a) is the unique value that identifies the agent a;
- Associated individual (I_a) is the tracked individual in the video which agent a trajectory should be based on. In our model, one agent has to be associated with one i in the video (from class A), and more than on agent can be associated with i from class B;
- Agent class (C_a) is the same class than individual I_a, i.e. ($C_a = C_i$). It determines if a should continue any unfinished trajectory (class A) or not (as discussed in Section 4.1); and
- Movement possibilities based on individual I_a (as defined in Equation 1).

When an agent is to be created, firstly, we detect how many are the unfinished trajectories (i.e. how many individuals i are in class A). It implies in how many agents a must be created to continue the real video. Such agents, belonging to class A, have to be created in the last positions of the individuals they are associated with in the video sequence, i.e. according to Equation 2:

$$\vec{X}_a^1 = \vec{X}_{I_a}^{ff}, \tag{2}$$

where ff represents the final frame of the video sequence. These agents can remain in the simulation indefinitely (static at $\vec{X}_a^f$ for all frames f), or they can keep walking, using the set defined in Equation 1.

Therefore, in order to allow different simulations generated from the same input video, we choose randomly the possible movement m to be applied for a certain agent a in frame f, as described in equation 3:

$$\vec{X}_a^{f+1} = \vec{X}_a^f + m, \tag{3}$$

where $m = rand(MOV_a)$.

4.3 Triggering Agents with new Trajectories

In case of agents that are not continuing any unfinished trajectory in the video (from class B), they should be created in the environment by the edges of the image in order to avoid that an agent suddenly appeared in the center of environment. We apply Equation 4:

$$\vec{X}_a^{f+1} = \vec{X}_a^f - rand(MOV_a), \tag{4}$$

until $\vec{X}_a^{f+1}$ reaches one of the edges. Note that the movement for each back step is randomly selected but belongs to possible movements from a (MOV_a). We can have multiple agents related to the same i trajectory and since we use random to select from the

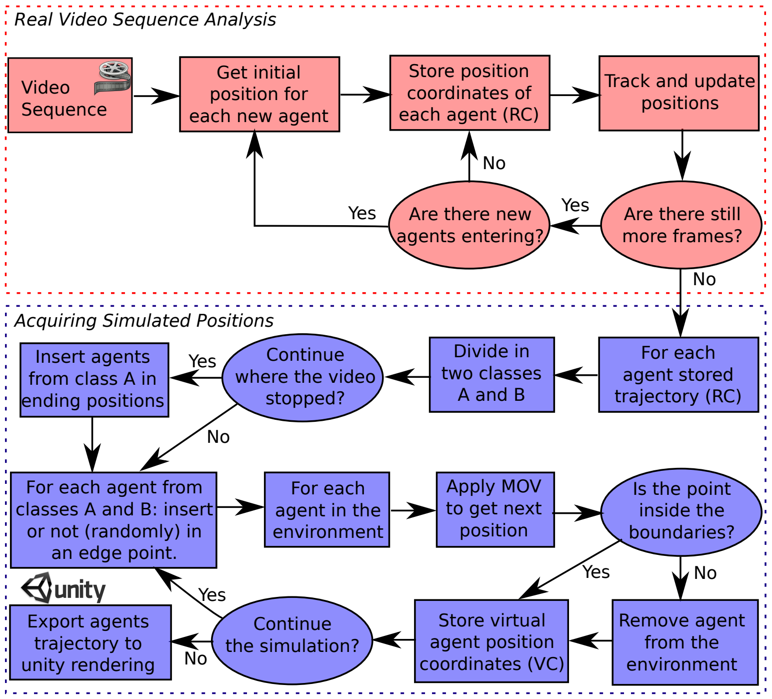
Real Video Sequence Analysis
Video Sequence
Get initial position for each new agent
Store position coordinates of each agent (RC)
Track and update positions
Are there still more frames?
Yes
Are there new agents entering?
Yes
No
No
Acquiring Simulated Positions
For each agent stored trajectory (RC)
Divide in two classes A and B
Continue where the video stopped?
Yes
Insert agents from class A in ending positions
No
For each agent from classes A and B: insert or not (randomly) in an edge point.
For each agent in the environment
Apply MOV to get next position
Is the point inside the boundaries?
Yes
No
Remove agent from the environment
Store virtual agent position coordinates (VC)
Continue the simulation?
Yes
No
Export agents trajectory to unity rendering
unity

Figure 3: Flowchart describing the entire process from Real Video Sequence Analysis (red) to Acquiring Simulated Positions (blue).

set MOV_a which is going to be applied, the resulting trajectories are not necessarily the same.

The trigger event occurs in regular intervals of 20 frames. In a given trigger event each individual i of class B has an equal chance to create an agent (and only one) based on his/her trajectory. The range of 20 frames is, in general, enough to avoid two agents to be created at the same edge point in almost the same time.

4.4 Removing Agents from the Environment

While in real video sequence the individual is removed when it is not traceable any more in the current video frame, the agents removal is a simple process in this work. In fact, we can identify when a virtual agent a leaves the room when, after applying a movement (Equation 1), its current position (x_a, y_a) is outside the environment's edges, i.e. Equation 5 is TRUE for any of conditions:

$$(x_a < 0) \vee (y_a < 0) \vee (x > img_{width}) \vee (y > img_{height}). \qquad (5)$$

Not necessarily all agents will be removed before the simulation ends. In fact, the simulation has not an end programmed, being able to keep continued.

4.5 Learning Trajectories and Simulating Virtual Agents

We can summarize the model process in two stages: first, the trajectory learning stage obtaining the pattern from video sequence analysis; second, the simulation stage to estimate the trajectory of the agents that are still in video or incoming new agents and performing a new real inspired trajectory. Figure 3 shows the entire process, from video sequence until exporting the trajectories for rendering on UNITY application. RC and VC are related respectively with real image coordinates and virtual world coordinates.

In 'Real Video Sequence Analysis' step, the individuals trajectories are tracked and stored. It occurs until the position of all individuals in each video frame has been obtained.

'Acquiring Simulated Positions' step is the main work contribution. It starts from the trajectories learned from real video sequence, then the set of trajectories/individuals (each one allocated to each agent a) is divided in two classes (as explained before). For video continuation purposes, agents in class A will be inserted in the first frame of simulation, after this step, agents based on any trajectory class (A or B) can be created, however starting always from an edge. Indeed, their creation occur in a random way, as simply stated in Equation 6.

$$CREATION_a = \begin{cases} rand(a) \leq K \rightarrow TRUE & \text{a is created} \\ rand(a) > K \rightarrow FALSE & \text{a is not created,} \end{cases} \qquad (6)$$

where $rand(a), K \in \Re |\ 0 \leq rand(a), K \leq 1$. The impact of K value is described in Section 5. Furthermore, increasing K will result in more agents in the environment, while decreasing K will generate less agents.

The simulation can run indefinitely, but in this work specifically, a 60 seconds was used, which corresponds to an amount of 1440 frames extra to real sequences. To visualize the simulations, all the agents trajectories are exported to Unity rendering. This process is explained in Section 5.

Table 1: Video Sequence Agents Data.

Video	Class A	Class B	Total of individuals	Frames	Time(s)
AE-01	7	5	12	230	10.00
AE-02	20	3	23	231	10.04
AT-01	8	4	12	401	17.43
AT-02	8	10	18	646	29.09
AT-03	6	4	10	364	15.83
BR-01	4	10	14	376	16.35
BR-02	22	0	22	150	6.52
BR-03	30	0	30	100	4.35
BR-04	29	0	29	50	2.17
BR-05	28	0	28	40	1.74
BR-06	10	4	14	340	14.78
BR-07	9	1	10	225	9.78
BR-08	12	2	14	200	8.70
CN-01	34	1	35	100	4.35
CN-02	23	5	28	100	4.35
CN-03	21	1	22	100	4.35
DE-01	29	1	30	200	8.70
DE-02	7	4	11	383	16.65
ES-01	14	6	20	220	9.57
FR-01	5	6	11	679	29.52
FR-02	5	1	6	759	33.00
JP-01	20	3	23	101	4.39
JP-02	27	1	28	100	4.35
PT-40	4	1	5	280	12.17
TR-01	29	12	41	187	8.13
UK-01	9	1	10	64	2.78
UKN-01	23	2	25	101	4.39
UKN-02	28	2	30	100	4.35
UKN-03	18	2	20	98	4.26

5 EXPERIMENTAL RESULTS

We developed an application using Unity to read a short video and based on the tracked information, run our learning method to generate NPCs trajectories. The experimental results have essentially two stages, the video analysis for learning trajectories and the simulation stage that generates a video continuation, mainly based on the insertion of new virtual agents in the environment.

As an input to our method, the Cultural Crowds [4] dataset was used. It consists of 33 crowded video clips from different countries. Some videos were manually acquired by the authors while others come from public datasets, like Collective Motion [22] and Data Driven Crowd [18], selected from Getty Images [20]. We used 29 videos in the training process as described in Table 1, where individuals from each class and duration time are presented.

Each of the 29 videos were used in training phase and a simulation of maximum 60 seconds was performed for each one. Therefore, the simulation should represent the same pattern of behavior (trajectories) as observed in the training video.

To compare obtained results in a same environment, the Figure 4

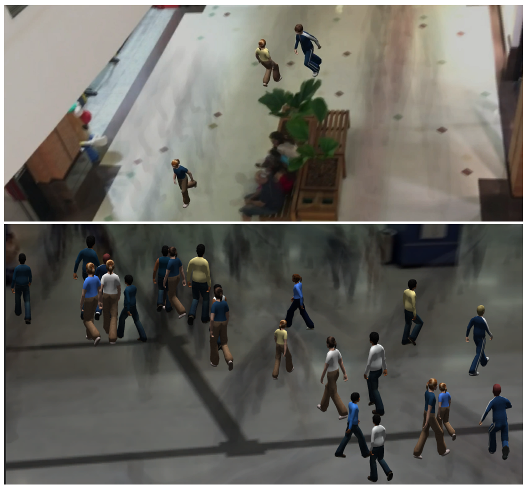
Figure 5: Simulations generated as a function of videos BR-07 (top) and UKN-02 (bottom).

shows some frames of an original sequence (on the top) and a simulated one (on the bottom). In such figure it is possible to note that the main trajectories are equivalent. In addition, the density of created agents can be adjusted using K value.

When the simulations end some quantitative values can be obtained, for example, how many agents were in the continuation process or still how many agents have been created based on extracted trajectories. These results differ as a function of the density observed in the input real video. For instance, Figure 5 illustrates two different simulations results generated as a function of 2 different videos (BR-07 and UKN-02) and same K value used ($K = 0.25$). As can be seen in Table 1, these two videos have respectively 10 and 30 individuals, and the amount of agents created is BR-07=23 and UKN-02=92. When we calculate the new individuals rate in the videos, we get values of 1.02 individuals per sec. (i/sec) in video BR-07 while 6.90 i/sec in video UKN-02. In the simulation we reach the values of 0.38 and 1.53 agents created per second. It happens because while in video BR-07 we have only 9 agents when the video ends (and the simulation begins), in video UKN-02 we have 28 already.

Other example (Figure 6) aims to illustrate the density variation generated as a function of K value. In this case, we used $K = 0.25$, 0.5 and 0.75 in video BR-02, and total generated agents are: 54, 104 e 129 for respective values of K (see Table 2) . Even if not all agents are simultaneously in the same moment in the video, it is easy to perceive quantitatively the variation of generated density.

Table 2 shows the obtained results for each tested video having three different values for $K = 0.25; 0.5; 0.75$. Column 'Cont.' shows the number of agents generated in class A at each video. It corresponds to the value informed in Table 1 'Class A' and shows that the continuation method provides continuation for all unfinished trajectories. As already illustrated for video BR-02, increasing the K value, for all videos, show an increment in the amount of generated agents.

Please notice that even the videos without individuals in class B (i.e. only individuals in class A are present), e.g. videos BR-02, BR-03 and BR-05) will generate agents that are not created to finish trajectories. As shown in Table 2, these videos generate respectively 32, 60 and 40 agents (if using $K = 0.25$) that do not serve for video continuation. Indeed, as previously described, such agents are created according to trajectories for individuals class A, however, they apply the requirement to find a position on the image edge to entry in the scene. In addition, the generated trajectories are not yet integrated to any crowd simulator, so it presents issues such as the collisions with obstacles and other agents, that are not avoided.

Note that generated simulations have 60 seconds while most part of training videos are small sequences between 2 and 16 seconds. Even in small sequences, our technique is able to reproduce acceptable simulation results that can be used in three-dimensional virtual environments.

6 Final Considerations

Generating trajectories for virtual agents using real video sequences is an alternative to create not repetitive paths for NPCs in games, what should enhance the immersion feeling of players, in general.

This paper presented a method to continue videos for a certain time keep going agents in the scene and creating others, according to the observed motion patterns. Results showed the generated

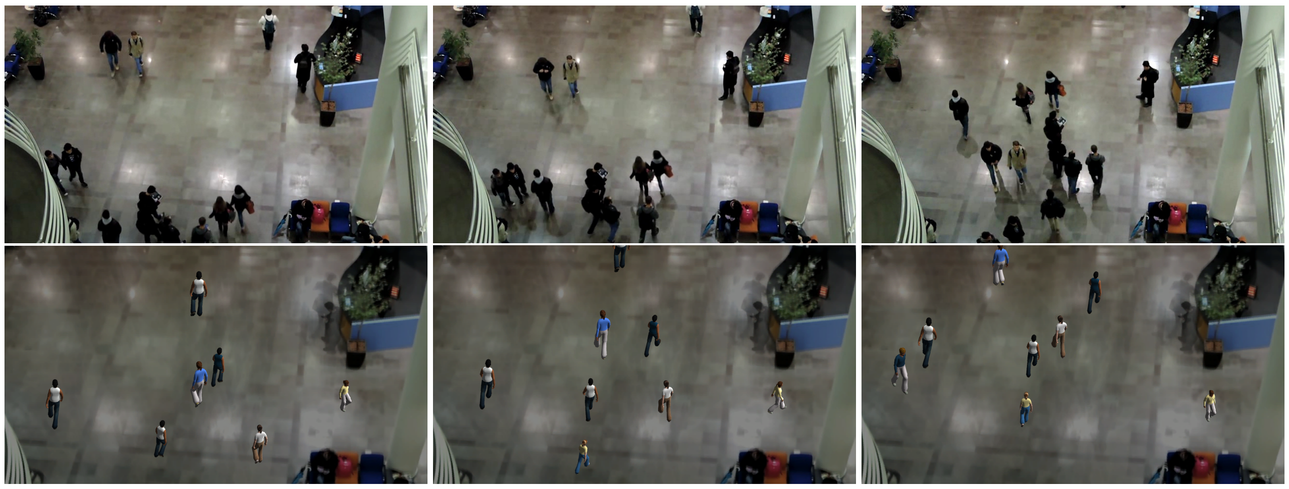
Figure 4: Original Video Sequence (on the top) and Simulation (on the bottom) based in video (BR-01).

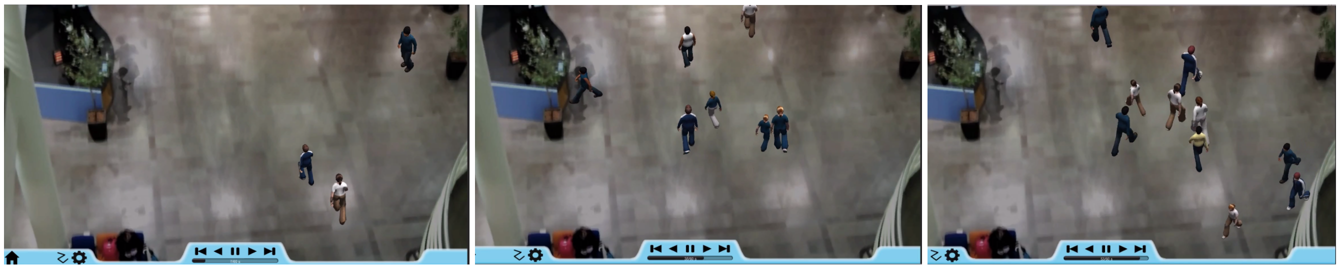

Figure 6: Simulations generated as a function of videos BR-02 and 3 values for $K = 0.25$, 0.5 and 0.75.

Table 2: Video Sequence Agents Data.

Video	Cont.	Created			Total of Agents		
$K(\%) \rightarrow$	All	25	50	75	25	50	75
AE-01	7	18	44	67	25	51	74
AE-02	20	42	87	119	62	107	139
AT-01	8	22	38	72	80	46	80
AT-02	8	28	60	123	36	68	131
AT-03	6	10	41	45	16	47	51
BR-01	4	23	52	71	27	56	75
BR-02	22	32	82	107	54	104	129
BR-03	30	60	108	180	90	138	210
BR-04	29	40	101	151	69	130	180
BR-05	28	53	112	148	81	140	156
BR-06	10	24	57	76	34	67	86
BR-07	9	14	49	55	23	58	64
BR-08	12	24	50	67	36	62	79
CN-01	34	65	130	91	99	164	125
CN-02	23	54	123	145	77	146	168
CN-03	21	36	78	125	57	99	146
DE-01	29	68	107	164	97	138	193
DE-02	7	15	50	64	22	57	71
ES-01	14	39	65	106	53	79	120
FR-01	5	23	36	66	28	41	71
FR-02	5	11	16	40	16	21	45
JP-01	20	35	76	127	55	96	147
JP-02	27	49	107	167	76	134	194
PT-40	4	7	18	24	11	22	28
TR-01	29	76	147	228	105	176	257
UK-01	9	16	29	58	25	38	67
UKN-01	23	43	95	147	66	118	170
UKN-02	28	64	117	154	92	145	182
UKN-03	18	35	73	92	53	91	110

virtual agents in existing trajectories and the agents density control, using K value.

As far as we know this area of NPCs control is still incipient in literature so, it indicates the originality of this work, mainly emphasizing the possibility of using simulation and computer vision in the context of virtual games.

As future work we intend to integrate the generated trajectories with a crowd simulation model, with the goal of making virtual people avoid collisions among them and with obstacles. Also, we want to work to numerically evaluate the resulted simulation and the input video. Some methods already existing in literature in order to compare crowds [15] and we want to investigate them to improve our evaluation process. Another future work is related to evaluation considering scripted and captured behavior. We believe we can achieve better results if we mix captured data with scripted behaviors. Anyway, some points have to be considered before a practical implementation in a game, such as collision avoidance, sources/destination and interrupt events.

7 Acknowledgments

The authors would like to thank FAPERGS, CNPq and CAPES.

References

[1] M. A. Akbar, M. Hariadi, W. Praponco, and M. S. N. Supeno. Multi behavior npc coordination using fuzzy coordinator and gaussian distribution. In *Intelligent Technology and Its Applications (ISITIA), 2015 International Seminar on*, pages 17–22, May 2015.

[2] E. F. Anderson. A npc behaviour definition system for use by programmers and designers. In *Proceedings of CGAIDE 2004 5th Game-On International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 203–207, 2004.

[3] J. Bins, L. L. Dihl, and C. R. Jung. Target tracking using multiple patches and weighted vector median filters. *MIV*, 45(3):293–307, Mar. 2013.

[4] R. Favaretto, L. Dihl, R. Barreto, and S. R. Musse. Using group behaviors to detect hofstede cultural dimensions. In *IEEE International Conference on Image Processing (ICIP)*, 2016.

[5] A. Fink, J. Denzinger, and J. Aycock. Extracting npc behavior from computer games using computer vision and machine learning techniques. In *2007 IEEE Symposium on Computational Intelligence and Games*, pages 24–31, April 2007.

[6] W. T. Freeman, K. Tanaka, J. Ohta, and K. Kyuma. Computer vision for computer games. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 100–105, Oct 1996.

[7] K. Fukunaga. *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990, 1990.

[8] M. Hu, S. Ali, and M. Shah. Learning Motion Patterns in Crowded Scenes Using Motion Flow Field. In *Proc. International Conference on Pattern Recognition (ICPR)*, 2008.

[9] T. S. Hussain and G. Vidaver. Flexible and purposeful npc behaviors using real-time genetic control. In *2006 IEEE International Conference on Evolutionary Computation*, pages 785–792, 2006.

[10] J. C. S. J. Junior, S. R. Musse, and C. R. Jung. Crowd analysis using computer vision techniques. *IEEE Signal Processing Magazine*, 27(5):66–77, Sept 2010.

[11] A. Khoo, G. Dunham, N. Trienens, and S. Sood. Efficient, realistic npc control systems using behavior-based techniques. In *Artificial Intelligence and Interactive Entertainment II*, pages 46–51, 2002.

[12] K. H. Lee, M. G. Choi, Q. Hong, and J. Lee. Group behavior from video: A data-driven approach to crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 109–118, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[13] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. *Computer Graphics Forum*, 26(3):655–664, 2007.

[14] S. H. Moon, M. Lee, S. Kim, and S. Han. Controlling npc behavior using constraint based story generation system. In *New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on*, pages 104–107, May 2010.

[15] S. R. Musse, V. J. Cassol, and C. R. Jung. Towards a quantitative approach for comparing crowds. *Computer Animation and Virtual Worlds*, 23(1):49–57, 2012.

[16] M. Paravisi, A. Werhli, J. C. S. Jacques Jr., R. Rodrigues, C. J. A. Bicho, and S. R. Musse. Continuum crowds with local control. In *Proceedings of the 2008 - Computer Graphics International 2008*, CGI'08, 2008.

[17] J. Pettré, J. Ondřej, A.-H. Olivier, A. Cretual, and S. Donikian. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 189–198, New York, NY, USA, 2009. ACM.

[18] M. Rodriguez, J. Sivic, I. Laptev, and J.-Y. Audibert. Data-driven crowd analysis in videos. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[19] D. Tosato, M. Spera, M. Cristani, and V. Murino. Characterizing humans on riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1972–1984, 2013.

[20] J. P. G. Trust. Getty provenance index database, May 2016.

[21] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE CVPR*, volume 1, pages I–511–I–518 vol.1, 2001.

[22] B. Zhou, X. Tang, and X. Wang. Measuring crowd collectiveness. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3049–3056, June 2013.

[23] C.-N. Zhou, X.-L. Yu, J.-Y. Sun, and X.-L. Yan. Affective computation based npc behaviors modeling. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, WI-IATW '06, pages 343–346, Washington, DC, USA, 2006. IEEE Computer Society.