Automated generation of various and consistent populations in multi-agent simulations

Benoit Lacroix and Philippe Mathieu

Abstract The variety and consistency of the agents behaviors greatly influences the realism in multi-agent simulations, and designing scenarios that simultaneously take into account both aspects is a complex task. To address this issue, we propose an approach to automatically create populations using sample data. It facilitates the designers tasks, and variety as well as consistency issues are handled by the generation model. The proposed approach is based on a behavioral differentiation model that describes the behaviors of agents using norms. To automatically configure this model, we propose an inference mechanism based on Kohonen networks and estimation distribution functions. We then introduce agents generators that can create a specified population, and are automatically configured by the inferred norms. The approach has been evaluated in traffic simulation, in association with a commercial software. Experimental results show that it allows to accurately reproduce the populations represented in sample data.

1 Introduction

The design of realistic scenarios in simulations is a crucial issue, as they play a key part in the users' immersion and the results validity. Moreover, the variety and consistency of the agents behaviors greatly influences the simulation outcomes. However, the scenarios design is often a complex task: introducing a high variety of behaviors into the simulation, while simultaneously avoiding any inconsistency, requires a careful configuration. Specific tools are therefore needed to assist the designer in these configuration tasks.

Such issues have been studied in the virtual reality field. For instance, Ulicny et al. [9] proposed specific tools for the designers, based on a *painting* metaphor. Us-

B. Lacroix · P. Mathieu
University of Lille, Computer Science Dept., LIFL (UMR CNRS 8022)
e-mail: lacroix.benoit@gmail.com ; philippe.mathieu@univ-lille1.fr

ing a *brush*, the designer can *paint* new pedestrians in the simulation, or behavioral characteristics on existing ones. This approach enables to easily create agents in the simulation and increase the variety. However, it remains focused on the graphical part of the simulation, and is unable to automatically build a population from sample data. Other works in crowd simulation [7] or traffic simulation [10] have focused on the introduction of variety in agents behaviors, through variations in their graphical or behavioral models. Nonetheless, these approaches involve complex interventions of the designer and have to be reproduced for each scenario creation. Regarding the simulation models, a learning-driven methodology to build them automatically was investigated in [2]. However, this methodology does not consider the configuration of existing models. Recent works [1] have addressed the automated validation of the multi-agent simulations results, by using statistical analysis. This latest approach is very complementary to ours, even though it rather focuses on the simulation validation than on the populations generation.

To address these shortcomings, we propose to automatically create agents populations in the simulations, while instantiating agents that display various and consistent behaviors. We based our approach on previous works [5, 6], where a behavioral differentiation model, representing agents behaviors using a *social norm* metaphor, was presented: each norm represents a set of agents sharing similar behavioral traits. In this paper, we propose a method to automatically configure this model using sample data, that can be either real measurements or simulated ones.

In more detail, this work advances the state of the art in the following ways:

- we present an original approach to automatically infer norms and behavioral parameters from sample data. This approach combines two unsupervised learning techniques: the inference of agents categories using self-organizing maps [4], and the estimation of the distribution function of the parameters. This method automatically configures the behavioral differentiation model by constructing the norms and their parameters.
- we present a novel technique to automate the generation of agents populations in simulations. This method is based on three elements: agents profiles, that describe the agents behavioral characteristics using norms; time slices, that associate a time dimension to sets of profiles; and generators, that group time slices and environmental parameters. Easily customizable and reusable, the generators provide a computer-based tool to assist designer during scenario creation.
- we combine the above to automatically configure the generators, in order to reproduce sample data. We apply this technique to traffic simulation, to create populations of vehicles. We evaluate our approach and show that the population that was created using the inferred generators is statistically similar to the original population, with an average difference of less than 3 %.

The rest of this paper is organized as follows. Section 2 briefly presents the characteristics of the behavioral differentiation model. In Section 3, we describe the inference mechanism. Section 4 presents the agents generators, and Section 5 describes the application to traffic simulation. Finally, Section 6 concludes and discusses the model further developments.

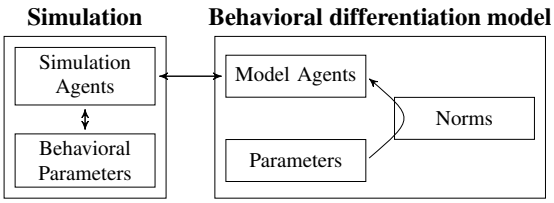


Fig. 1 The designer specifies *parameters* and *norms*, which provide a behavioral patterns used to instantiate the *model agents*. The *model agents* are used to assign parameters values to the *simulation agents* and control the *norm* respect.

2 Behavioral Differentiation Model

In this section, we briefly introduce the main characteristics of the behavioral differentiation model. This model enables to instantiate various and consistent behaviors for the agents, and to produce behaviors representative of real world situations [6]. Moreover, it was developed in an industrial context, and no modification of the simulation source code is needed to integrate it in commercial softwares.

In this model, the behaviors of the *simulation agents* are described using a *social norm* metaphor. The *norms* represent behavioral patterns that specify agents behaviors. They allow to generate the parameters values of the agents at their creation, and to control their conformity at runtime. During scenario creation, the designer specifies *norms* and *parameters* in the behavioral differentiation model (Fig. 1). *Model agents* are middlemen between the model and the simulation: at the creation of the *simulation agents*, a *norm* is instantiated in a *model agent*; the values of this *model agent* are then sent to the corresponding *simulation agent*; finally, at runtime, the *model agents* are used to control the conformity of the *simulation agents* values.

The *Parameters* represent the *behavioral parameters* of the *simulation agents* in the model. A *Parameter* which reference parameter is null is called a *root parameter*.

Definition 1. A *Parameter p* is a tuple $(p_{\text{ref}}, \mathscr{D}_p, v_{d_p}, g_p, f_p)$ defined by:

- $p_{\text{ref}}(p)$ a reference parameter,
- $\mathscr{D}_p$ a finite definition domain, with if $p_{\text{ref}} \neq null$, then $\mathscr{D}_p \subseteq \mathscr{D}_{p_{\text{ref}}}$,
- $v_{d_p} \in \mathscr{D}_p$ a default value,
- g_p a probability distribution over $\mathscr{D}_p$, with by default g_p a uniform distribution,
- $f_p : \mathscr{D}_{p_{\text{ref}}} \mapsto [0,1]$ a distance function that allows to compute the gap between a value and p definition domain, with by default $\forall x \in \mathscr{D}_{p_{\text{ref}}}$, $f_p(x) = 0$.

The *Norms* specify the agents behaviors. They represent a behavioral pattern, used during agents creation and conformity checks. A *Root norm* holds all the *root parameters*, and is used to check agents conformity with their specification.

Definition 2. A *Norm N* is a set $\{N_{\text{ref}}, \mathscr{P}_N, \mathscr{Q}_N, \tau_N, \delta_{\max_N}\}$ with:

- $N_{\text{ref}}(N)$ a reference norm,
- $\mathscr{P}_N$ a finite set of parameters p,
- $\mathscr{Q}_N$ a set of properties,
- τ_N a violation rate of the norm, which describes the proportion of violating behaviors, with $\tau_N \in [0,1]$. By default, $\tau_N = 0$,

- $\delta_{\max_N}$ a maximal gap to the norm, which describes the tolerance towards norm violations, with $\delta_{\max_N} \in [0,1]$. By default, $\delta_{\max_N} = 1$.

Finally, the *Model Agents* represent the instantiation of a *norm*: they include all the *parameters* of their reference *norm*, and associate a value to each of them.

Definition 3. A *Model Agent* a_m is a set $\{N_{a_m}, \mathscr{C}_{a_m}\}$ with:

- N_{a_m} a reference norm,
- $\mathscr{C}_{a_m} = \{(p, v_p),\ p \in \mathscr{P}_{N_{a_m}}\}$ a set of pairs of parameters and associated values.

Example 1. In a traffic simulation, each driver is characterized by a set of behavioral parameters, like the maximal speed or the security distance. The definition of such *behavioral parameters* in the behavioral differentiation model can result in the following *Parameters*:

- the maximal speed v_{max}, defined by $p_{\text{ref}} = null$, $\mathscr{D}_p = [0,200]$ km/h, $v_{d_p} = 100$ km/h,
- the safety time t_s, defined by $p_{\text{ref}} = null$, $\mathscr{D}_p = [0.1,3]$s, $v_{d_p} = 1.5$s, which represents the security distance d ($d = t_s \cdot v$),
- the normal maximal speed on highway v_{normal}, defined by $p_{\text{ref}} = v_{max}$, $\mathscr{D}_p = [110,130]$ km/h, $v_{d_p} = 120$ km/h, g_p the normal distribution described by a mean value $\mu = v_{d_p}$ and variance $\sigma^2 = 5$ truncated at $\mathscr{D}_p$ bounds,
- the normal safety time t_{normal}, defined by $p_{\text{ref}} = \text{t}_\text{s}$, $\mathscr{D}_p = [1,2.5]$s, $v_{d_p} = 1.5$s.

The root norm N_{root} is defined by $\mathscr{P}_{N_{\text{root}}} = \{v_{max}, t_s\}$. The norm N_{normal}, defined by $N_{\text{ref}} = N_{\text{root}}$, $\mathscr{P}_N = \{v_{max,normal},\ t_{normal}\}$ and $\mathscr{Q}_N = \emptyset$, represents a normal driving style. *Simulation agents* created from the *model agents* instantiated from N_{normal} will therefore adopt this particular driving style.

This model is associated to an algorithm inspired from fuzzy path following techniques [8], and further presented in [5]. It enables to instantiate various behaviors for the agents within the same norm.

3 Automated Configuration of the Model

In this section, we present a method to automatically configure the behavioral differentiation model using sample data. The objectives are to ease the designers work and facilitate the use of the model. To avoid any need for user supervision and preserve genericity, we chose to combine two unsupervised learning techniques: Kohonen networks [4], also called self-organizing maps, and distribution function estimation.

Algorithm 1 presents the procedure used. First, a Kohonen network of rectangular topology is created. To minimize user configuration, the number of clusters (ie. of neurons) is dynamically computed, function of the dimension d of the input vectors: here, $k = (d+1)^2$. For each of the k clusters, a *norm* N_k and d *parameters* $p_{k,i}$

Algorithm 1 Automated creation of norms

Require: a set of inputs $\mathscr{E} = \{e\}$ with d the dimension of the input vectors e
1: create the Kohonen network $\mathscr{K}$ of rectangular topology with $k = (d+1)^2$ neurons of weights $W_i = (w_{i,j})$ ($i \in [1,k]$ and $j \in [1,d]$) ; train $\mathscr{K}$ with the set of examples $\mathscr{E}$
2: **for all** $i \in [1,k]$ **do**
3: create a *Norm* N_i such that $Q_{N_i} = \emptyset$, $\tau_{N_i} = 0$, $\delta_{\max_{N_i}} = 1$, and $N_{ref}(N_i) = N_{\text{root}}$
4: **for all** $j \in [1,d]$ **do**
5: create a *Parameter* $p_{i,j}$
6: save the weight value $w_{i,j}$ of the neuron i as the default value of $p_{i,j} : v_d(p_{i,j}) \leftarrow w_{i,j}$
7: **end for**
8: **end for**
9: **for all** $e \in \mathscr{E}$ **do**
10: classify the example e using the network $\mathscr{K}$. Let W_i be the weights of the triggered neuron
11: **for all** $j \in [1,d]$ **do**
12: if $w_{i,j}$ is greater than the maximum or lower than the minimum of $\mathscr{D}_{p_{i,j}}$, update the corresponding bound of the domain
13: add the value e_j to the distribution estimator of the *Parameter* $p_{i,j}$
14: **end for**
15: **end for**

($i \in [1,d]$) are created. This *norm* represents the inferred cluster, and the *parameters* represent the different dimensions of the input data.

The set of example $\mathscr{E}$ is then used to train the Kohonen network. The outputs are the values vector W_k of the k neurons. By construction, for each k, each value $w_{k,i}$ of W_k matches the parameter $p_{k,i}$ of N_k. These values define the default value of each of the *parameters*: $\forall i \in [1,d],\ v_d(p_{k,i}) = w_{k,i}$.

We still have to determine the definition domain of each of the *parameters*, as well as the associated probability distribution. To compute these elements, the Kohonen network trained during the previous steps is used to classify the sample data. For each k, $\mathscr{E}_k \subset \mathscr{E}$ is the set of inputs in cluster k. The bounds of the definition domain of the parameter $p_{k,i}$ are the extremes values taken by this dimension in $\mathscr{E}_k$: if $\mathscr{D}_{p_{k,i}} = [a_i, b_i]$, then $a_i = \min_{\mathscr{E}_k}\{e_i\}$ and $b_i = \max_{\mathscr{E}_k}\{e_i\}$. We combine this step with an estimation of the distribution function representing these data. Without loss of generality, we suppose that the data follow a normal distribution function. Using a method based on the maximum likelihood, we estimate the normal distribution parameters.

This procedure automatically creates a set of *norms* representing the sample data. It provides a method to easily parameterize a model to reproduce observed situations. For instance, in crowd or traffic simulations, data can be recorded in the real world, and can then be used to infer a set of norms. The same method can be used with data recorded during a simulation run. We can then reproduce an experimental setting by creating a situation similar to the recorded one.

4 Generation of Agents Populations

In this section, we describe a method to easily populate a database with agents, while specifying precisely the composition of the population. To achieve this goal, the proposed tool combines profiles, time slices and generators.

A *Profile* is associated to a *Norm*, to specify the behavioral pattern of the agents it will create. It also includes a set of properties: in traffic simulation, one of these can be an itinerary, to create origin/destination traffic demands. The set of *profiles* is noted $\mathscr{P}_r$.

Definition 4. A *Profile* p is a defined by:

- a norm N_p,
- a set of characteristics $\mathscr{Q}_p$.

A *Time Slice* holds a set of *profiles*. It is active permanently or during a specific time interval, and specifies the properties of the population created during that period: the profiles that will be used, the proportion of each of them, and the frequency of agents creation. The set of *time slices* is noted $\mathscr{T}_s$.

Definition 5. A *Time slice* t is defined by:

- a duration d_t, with $d_t = [t_{start}, t_{stop}]$. If $t_{start} = t_{stop} = 0$, t is permanently active,
- a set $\mathscr{P}_t$ of profiles, associated to the relative percentage of this profile in the population: $\mathscr{P}_t = \{(p, pc), p \in \mathscr{P}_r, pc \in [0,1] \text{ and } \sum_{p \in \mathscr{P}_t} pc = 1\}$,
- a frequency of generation f_t (in s^{-1}).

Finally, a *Generator* includes a set of *time slices* and specifies the position at which the agents will be generated in the environment. Only one *time slice* may be active at the same time, and, by default, the agents are created at a random position in the environment. The set of *generators* is noted $\mathscr{G}$.

Definition 6. A *Generator* g is defined by:

- a set of time slices $\mathscr{T}_g$, with $\forall t_1, t_2 \in \mathscr{T}_g^2, t_1 \neq t_2 \Rightarrow d_{t_1} \cap d_{t_2} = \emptyset$,
- a function $f_g : \mathscr{A} \rightarrow \Re^3$ associating a position in space to an agent.

Example 2. In addition to the *norm* N_{normal} defined in Example 1, we suppose that a *norm* $N_{aggressive}$ describing aggressive drivers, adopting higher maximal speeds and lower security distances, is defined. We specify two *time slices* representing the traffic characteristics at different hours during the day:

- t_1 represents the rush-hour, when drivers are aggressive and the circulation dense: $d = [7,9]$h, $\mathscr{P}_{t_1} = \{(p_{aggressive}, 0.2), (p_{normal}, 0.8)\}$ and $f = 1$ (dense flow of 3600 veh/h)
- t_2 represents the remaining of the morning period, when only normal drivers are present: $d =]9,12]$h, $\mathscr{P}_{t_2} = \{(p_{normal}, 1.0)\}$ and $f = 0.2$ (flow of 720 veh/h)

We then define the *generator* g_1 with $\mathscr{T}_{g_1} = \{t_{s_1}, t_{s_2}\}$ and $f_{g_1} : \mathscr{A} \rightarrow (0,0,0)$. It creates a population following the specified characteristics at the position (0,0,0).

Algorithm 2 Creation of the agents by the generators.

Require: t the current timestep, $\mathscr{G}$ the set of generators
for all $g \in \mathscr{G}$ **do**
  if $\exists t \in \mathscr{T}_g$, such that $t \in d_t$ **then** {the time slice t is active}
    $\alpha \leftarrow$ uniform_random($[0,1]$) ; $\beta \leftarrow 0$
    for all $(p, pc) \in \mathscr{P}_t$ **do**
      if $\beta \leq \alpha < \beta + pc$ **then** {select this profile}
        generate an agent using the behavioral differentiation model and norm N_p
      end if
      $\beta \leftarrow \beta + pc$
    end for
  end if
end for

The algorithm 2 describes how the agents are created by the *generators*. At each time step, we check for each *generator* if it includes an active *time slice*. If so, we randomly select one of the *profiles* held in this *time slice*, using the probability *pc* to balance this choice. An agent matching this *profile* is then automatically created using the behavioral differentiation model and the specified *norm*.

Moreover, by combining the generators with the inference mechanism presented in Section 3, generators can be automatically configured. To do so, a *profile* is created for each of the inferred norms. This set of *profiles* is associated to a single *time slice*, permanently active, and the percentage of each profile is set at the corresponding proportion of agents matching this norm in the sample data set. Finally, a *generator* holding this *time slice* is created. The user only has to specify the position he wants the agents to be created at, if a random position does not suits his needs.

The agents generators enable to easily create a varied and consistent population in the simulation.

5 Experimental Evaluation

In this section, we apply the presented method to traffic simulation in driving simulators. Driving simulators are used for instance in the automotive industry for design studies, or to develop driving aid systems. Our study is based on the software developed and used at Renault, SCANeR™ [3], which is co-developed and distributed by Oktal. In SCANeR™, the design of scenarios involves complex configuration steps: each vehicle has to be individually created, and each of its behavioral parameters manually modified. We therefore integrated the proposed approach with SCANeR™, to automate this configuration. To validate the approach, we propose here a first evaluation based on data recorded from the simulation.

5.1 Experimental Protocol

The evaluation is based on a database representing an highway, on a 11 km long section. In SCANeR^{TM}, the behavior of the drivers is influenced by different behavioral parameters: the maximal speed, the safety time, an overtaking risk, and factors denoting the respect of speed limits, road signs and priorities. We simulated induction loop detector in the database at kilometer 6.6, and recorded the vehicles data. In the real world, such detectors provide elements about the vehicles speed and safety times. Therefore, we based this analysis on these two parameters. To evaluate the proposed approach, we used the following protocol:

1. generation of a population of vehicles in the database, using pre-configured generators. These generators create a flow of 3000 vehicles per hour, with 10 % of cautious drivers, 10 % of aggressive ones, and 80 % of normal ones,
2. recording of the data of the vehicles crossing the detector, during one hour after the simulation has reached a steady state. This provides the data set $\mathscr{E}_1$,
3. norms inference based on $\mathscr{E}_1$ and construction of the associated generator,
4. generation of a population using the generator constructed in step 3,
5. recording of the data, which provides the data set $\mathscr{E}_2$,
6. comparison of the two populations using statistical analysis.

5.2 Results

After the generation and the recording of a first population of vehicles (steps 1 and 2 of the protocol), we obtain the sample data set $\mathscr{E}_1$ (3471 examples). Using the Algorithm 1, we automatically infer 9 different norms. Figure 2 graphically represents these norms, using a diamond shape placed at the coordinates of the default value of their parameters. Around each of these points, a rectangular shape represents the definition domain of the parameters, and the grayscale filling of these rectangles denotes the proportion of agents belonging to this norm in the population. Then, we create a generator g based on the inferred norms. This generator include a single time slice, permanently active, associated to the frequency $f = 3471/3600$ (recorded flow of 3471 veh/h). This time slice includes 9 profiles, each one associated to one of the inferred norms. Their proportion is set to the relative value of the population matching this norm in the sample data set $\mathscr{E}_1$. Another simulation is then launched, where the vehicles are created using the generator g at the same position as in step 1. The vehicles data are recorded, which provides the data set $\mathscr{E}_2$ (3487 examples).

To statistically compare the two populations represented in the data sets $\mathscr{E}_1$ and $\mathscr{E}_2$, we infer the norms using the sample data $\mathscr{E}_2$, and compare these with the norms obtained with $\mathscr{E}_1$ in step 3. The results are the following (no significant variation was observed among different simulation runs). When comparing each cluster with the closest one (Fig. 3), the default values of the parameters do not differ of more than 2.3 % (on average 0.3 % and 0.8 % for speed and safety time, respectively).

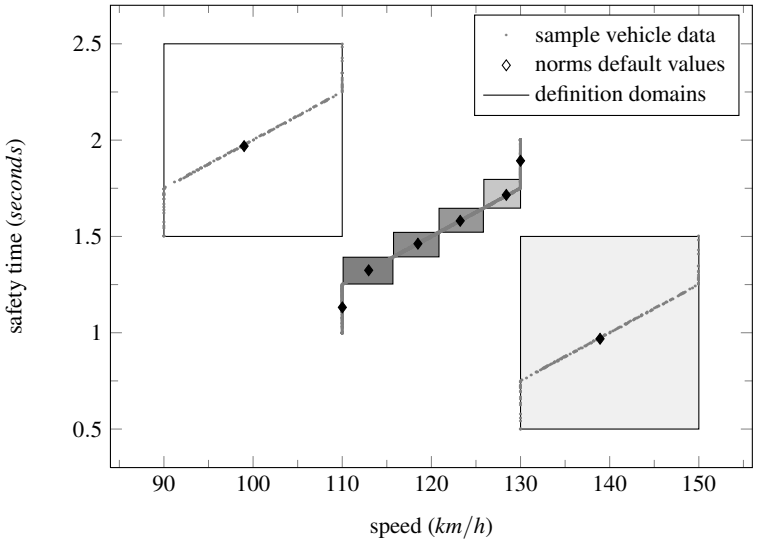
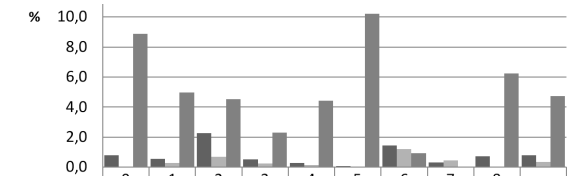


Fig. 2 The *norms* are represented by the default values and the definition domains of their *parameters*. The gray scale of the rectangles representing the definition domains denotes the proportion of examples belonging to this norm in the input data set.

The bounds of the definition domains remain within a 1.7 % limit, on average, with a maximum of 8.32 %. The repartition of the population in the closest clusters varies up to a maximum of 10.2 % (average at 4.7 %). However, this shift in the population repartition produces an average speed of the vehicles of 102.5 km/h (population $\mathscr{E}_1$), instead of 113 km/h (population $\mathscr{E}_2$).

Fig. 3 Comparison of the differences between the characteristics of each the 9 norms inferred from $\mathscr{E}_1$ and from $\mathscr{E}_2$. Parameters vary from less than 2.3 %, and the population affected to each of the norm from less than 10.2 % (4.7 % on average).



	0	1	2	3	4	5	6	7	8	avg
safety time	0,8	0,6	2,3	0,5	0,3	0,1	1,4	0,3	0,7	0,8
speed	0,0	0,3	0,7	0,2	0,1	0,0	1,2	0,5	0,0	0,3
population	8,9	5,0	4,5	2,3	4,4	10,2	0,9	0,0	6,2	4,7

The clusters inferred from the first and second populations are therefore very similar, which shows the robustness of the proposed mechanism. Moreover, the population created using the inferred norms expresses the same behavioral characteristics as the sample population. However, the difference in the repartition of the population in the different clusters produces a more “careful” population than the initial one. Those results show that the proposed approach enables to automatically create generators that reproduce a population statistically close to the initial one.

6 Conclusion

We have presented a method to automatically generate populations of agents from sample data. Based on a description of agents behaviors using a *social norm* metaphor, this method combines two elements. First, it infers the configuration of the behavioral differentiation model using Kohonen networks and an estimation of the parameters distribution functions. Second, generators automatically create populations of agents that display various and consistent behaviors, by taking advantage of the behavioral differentiation model. We combined these two elements and applied them to traffic simulation in driving simulators. After integrating the model into a commercial simulator, we showed that the proposed approach enabled us to create agents population statistically close to the sample data sets.

Additionally, this approach can be used in any simulation where the agents behaviors can be defined as parameters and norms, e.g. pedestrians behaviors in crowd simulations.

Future works will evaluate the approach with data recorded from the real world, and to improve the inference mechanism to automatically minimize the number of inferred norms. Finally, the evolution of the norms during time could lead to the definition of different time slices, associated to specific norms sets. This would provide another interesting tool for the user, and bridge the gap with current works on the automated observation of complex simulations.

References

1. P. Caillou. Automated multi-agent simulation generation and validation. In *Int. Conf. on Principles and Practice of Multi-Agent Systems*, pages 398–413, 2010.
2. R. Junges and F. Klugl. Evaluation of techniques for a learning-driven modeling methodology in multiagent simulation. In J. Dix and C. Witteveen, editors, *Multi Agents System Technologies*, volume 6251 of *LNAI*, pages 185–196. Springer, 2010.
3. A. Kemeny. A cooperative driving simulator. In *International Training and Equipment Conference*, pages 67–71, London, UK, 1993.
4. T. Kohonen. *Self-Organizing Maps*. Springer, 1995.
5. B. Lacroix, P. Mathieu, and A. Kemeny. Generating various and consistent behaviors in simulations. In *Int. Conf. on Practical Applications of Agents and Multi-Agent Systems*, pages 110–119, Salamanca, Spain, 2009.
6. B. Lacroix, P. Mathieu, and A. Kemeny. The use of norms violations to model agents behavioral variety. In J. Hubner and et al., editors, *Coordination, Organizations, Institutions and Norms in Agent Systems IV*, volume 5428 of *LNCS*, pages 220–234. Springer, 2009.
7. J. Maim, B. Yersin, and D. Thalmann. Unique character instances for crowds. *IEEE Computer Graphics and Applications*, 29(6):82–90, 2009.
8. C. W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference*, pages 763–782, San Francisco, USA, 1999.
9. B. Ulicny, P. de Heras Ciechomski, and D. Thalmann. Crowdbrush: Interactive authoring of real-time crowd scenes. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Grenoble, France, 2004.
10. S. Wright, N. J. Ward, and A. G. Cohn. Enhanced presence in driving simulators using autonomous traffic with virtual personalities. *Presence*, 11(6):578–590, 2002.