Learning by implicit imitation in virtual worlds

Ahmad Abdul Karim
Spirops - Saara team, LIRIS
University Claude Bernard Lyon 1, France
ahmad.abdul.karim@spirops.com
http://liris.cnrs.fr

Cédric Sanza
Vortex team - IRIT
University of Toulouse, France
sanza@irit.fr
http://www.irit.fr

Abstract

In this paper, we present a system that generates the behavior of pedestrians in a virtual town without explicit programming. This work consists in the creation of a dynamic set of rules based on the imitation of the actions of a user.
During the first stage, the user pilots the avatar in the environment and the system records all the couple sensors-actions. In the second stage, the *version space* algorithm is used to compress the database into a set of rules composed of a set of intervals that represents the sensor part, and a discrete action for the effector part. In the last stage, the rules are directly injected in a *learning classifier system* that controls the autonomous character. Thus, the entity can simply use the rules and generate new ones to cope with new situations.
Different scenarios give encouraging results with short recording stages. The main rules are generated from imitation and the secondary rules come from the ability of the learning classifier system to adapt the rules to the environment.

Keywords: behavioural simulation, imitation, crowd, classifier systems

1. Introduction

In crowd simulations, many researches focus on collision detection and path planning. Few works are interested in intelligent behaviors of the characters because it is a very complex task. The main techniques for designing such problems are *Top–Down* methods and *Bottom–Up* methods.
In the Top–Down methods, the designer uses the complete knowledge of a given problem, with all the possible situations, actions and rewards. The most popular techniques are based on graphs:

- finite state machines: mainly used in video games, but rather difficult to construct,
- decision tree: automatically generated by using the concept of entropy,
- artificial neural networks: created from a set of examples, can give a response to an un-learned situation.

These three methods are robust in deterministic environments but a new situation can lead to completely reconstruct the graph by using the complementary knowledge. Once the system is online, the graph remains static and cannot evolve to deal with complex and unpredictable environments.
In the *Bottom–Up* Methods, the designer gives the necessary tools to the system to solve the problem, without explicitly giving the solution. The system needs to combine these tools and use them properly to find a possible solution. Exploitation and exploration are abilities that enable such systems to use the knowledge already learned and to continuously try to find better ones. The main techniques are:

- genetic algorithms: a set of randomly generated solutions evolves thank to a fitness function to finally give the best solution,
- learning classifier systems: based on genetic algorithms, it consists in a set of rules that converges towards an efficient global behavior.

In interactive simulations, the main problem of these methods is the time needed to find the solutions which can be rather long,
In this paper, we propose to use the combination of a bottom-up method and a top-downd method (*version space* [1] and *learning classifier systems* [2]) to define the behavior of the characters. Our idea is based on the imitation of a real user to produce a set of evolving rules to manage pedestrians in a city.
Section 2 presents the concept of imitation. Section 3 describes the experiment. Results are presented in section 4. The last section concludes this paper with future works.

2. Imitation

Imitation is a really powerful tool specially used by the human brain to increase its knowledge and to discover solutions. The imitation strategy highly reduces the search space for an appropriate solution. It is an efficient way of enhancing machine learning in multi-agent systems: the agent can use the knowledge of the past cooperative teachers, or other different agents to operate in the same environment.

In machine learning by imitation there are two roles which are the observer and one or several mentors, and two main type of imitation: explicit and implicit imitation.
In explicit imitation, the mentors take the role of the teachers, and their goal is to make the observer imitate them, by directly teaching the right set of decisions and actions. In this case the main assumptions are that the mentors are cooperative and ready to share their knowledge with the observer, and that the system is capable of incorporating this kind of direct communication [3,4].
In implicit imitation, there is no direct communication between the observer and the mentors, and the agents are not forced to play the role of the teacher explicitly, on the contrary, the observer tries to simply learn by copying the behavior of other agents. In this case, we can imagine that it is not possible for a mentor to alter its behavior to teach the other agents, or it could be unwilling to do that because they are in a competitive situation [5,6,7,8].

There are many differences between the two strategies. For example in implicit imitation the agent is not forced to perfectly imitate the mentors. He can use the observations he made to enforce his own system, while in explicit learning, the observer is expected to be whatever the mentors wants him to be. Another difference is that in implicit imitation the goals of the mentors can differ from those of the observer and that is why the imitation cannot be exactly identical. In this case, the observer needs to imitate the more interesting behaviors concerning its proper objectives.

We need to distinguish between two settings: homogenous settings where the set of action and abilities between the observer and the mentors are the same, in this case the mapping between the mentors and the observer actions and decisions is straight forward while in heterogeneous settings, there might be differences, which could lead to odd situations where the mentor is capable of interacting with the environment differently or he has different capabilities. The observer needs then to make adjustments during the imitation process to adapt everything to his own model.

In any kind of imitation there are 3 main steps: the recording stage, the machine learning stage and the re-playing stage.

The information recorded in the first step depends on the abilities of the agent. The more complex is the simulation, the more complex is the kind of data recorded. In a simple grid, the information is limited to the eight surrounded cells [5,6]. In a LAN-party video game, the recorded information is huge and heterogeneous [7].

The machine learning stage is the more complex part of the imitation. Several methods are used to compress the database into different formats like decision tree [8], set of rules [6],

or neural networks [7] which are directly used during the replaying-stage to animate the characters.

All the works where imitation was used show that the behaviour of the agent is obtained with few effort compared to methods like scripting where the designer has to write the correct code.

3. Experiment

3.1 The environment

The test environment is an in-house developed 3D city simulation, with different kind of objects like vehicles, pedestrians, traffic lights and different kind of surfaces like sidewalks, crossroads, streets (figure 1). We use 3D models to represent these objects and surfaces plus we also use Motion Capture for the pedestrian's movements.

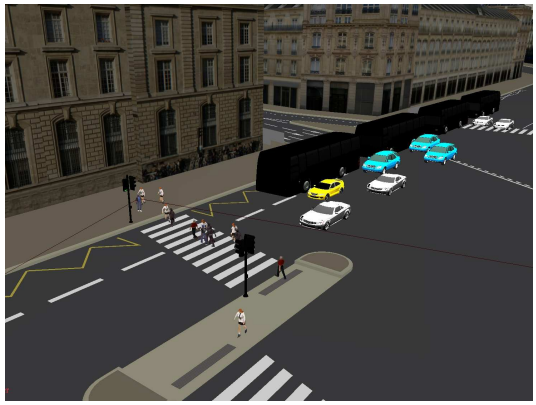

Figure 1: "Place de la république" in Paris

This simulation is as real as possible and this was achieved by using physics to manage the force that drives most of these objects, like the interactions between the pedestrians and the environment (going up the stairs, bumping with each others), the movement of the vehicles (suspensions, breaking forces), by using the open source physics engine Bullet.

3.2 The system of imitation

The system of imitation we build is implicit. It is based on the Métivier concept of: *guidance interaction* [6], where, at any moment of the simulation, the user can connect to the environment and control his avatar, and when he disconnects, the system will take control of this *Avatar* and the imitation process will begin. We have only one *mentor* who is the *observer* at the same time, and they both have the same capabilities *(homogenous setting*) and share the same objectives.

During the recording stage, the perception of the character (cone of vision) is recorded as a set of attributes: id, type, name, orientation of the object, distance and angle between the object and the avatar, and specific parameters (speed for the vehicles…). All the attributes (real, integer, Boolean, string) are unified to the [0..1] interval in order to work with fuzzy rules.
The machine learning stage consists in four sub-stages:

- the data filtering reduces the information recorded in the database by not processing the current frame information when no changes occur in the environment
- the version space algorithm creates the set of rules. This algorithm has been extended to avoid the loss of data encountered when contradictions appear. In our system, a second rule is created instead of deleting the first one
- the generation of the forces of the rules based on two parameters : the number of the observations that the rule covers, and the duration of these observations
- the generalization of the rules by merging the most specific ones.

The re-playing stage adds all the rules into a learning classifier system [G] that runs with specific fitness functions. The reward penalizes the rules that generate collisions or bad behaviors like crossing the street at the wrong way.

4. Results

We have tested our system in scenarios where the mentor crosses a street according to the

traffic light, the vehicles and the other pedestrians.

The first Scenario is a simulation without vehicles and pedestrians. The traffic lights are the only active object. In this case, two or three street crossing are necessary to perform the learning and creating the proper LCS. Based on this LCS, and during the re-playing stage, the avatar then crosses the street when the light is green and stops when it is red.

In the second scenario, we have added some vehicles in the street. The mentor crosses the street when the light is green. For pedestrians while it is red for cars. In the replay, the avatar correctly crosses the street but we sometimes notice a new interesting behaviour, the avatar crosses the street when there is no car, whatever the color of the lights. This behaviour comes from the number of objects perceived in the scene. The number of vehicles, a lot more significant than the number of traffic lights, is taken into account by the rule as a parameter for crossing the street: the avatar learns to cross when there is no car in the crosswalk, instead of crossing according to the traffic light. This rule remains persistent in the base as it does not trigger collision with the cars.

In the last scenario, we have added pedestrians that crosses the street when the light is green. As pedestrians are in average more numerous than cars, the avatar learn to cross when other pedestrians cross the street.

This three simulations show that the system works but it could be optimized by a better generalization of the rules, enabling it for example to focus on the traffic light instead of the other objects. The classifier system brings a part of the solution as it evaluates the rules and ensure a coherent simulation.

5. Conclusion

We have presented a system that enables to generate the behavior of a character in a city from the imitation of a real user. The main interest of our method is the symbiosis between a bottom-up and a top-down method. Even if the rules coming from the imitation are no perfect, the learning classifier system can manage every situation by adapting the rules to the environment.
Applying such a system to a crowd implies the imitation of several users. The next step is the interpolation of the imitation of two users in order to generate a set of various behaviours.

References

[1] T. M. Mitchell, *Version Spaces: An Approach to Concept Learning*. PhD thesis, Stanford University, 1978

[2] S. Wilson, *ZCS: A zeroth level classifier*. Evolutionary computation Vol2, 1994

[3] C. Atkeson, S. Schaal *Robot learning from demonstration*, Proceedings of the Fourteenth International Conference on Machine Learning, pp. 12-20. Nashville, 1997

[4] S. Whitehead, *Complexity analysis of cooperative mechanisms in reinforcement learning*, Proceedings of the Ninth National Conference on Artificial Intelligence, pp. 607-613, Anaheim, 1991

[5] B. Price, *Accelerating Reinforcement Learning with Imitation*, Ph.D. thesis, University of British Columbia, 2003

[6] M. Métivier, *Méthodes évolutionnaires et apprentissage : Apprentissage par imitation dans le cadre des systèmes de classeurs*, Ph.D. thesis, University René Descates, Paris, 2004.

[7] C. Thurau, G. Sagerer, C. Bauckhage, *Imitation learning at all levels of Game-AI*. International Conference on Computer Games: Artificial Intelligence, Design and Education, pp. 402-408, 2004

[8] W. Tambellini, C. Sanza, *Behaviors Generation with Artificial Intelligence in Video Games*. *International Digital Games Conference*, Portalegre, Portugal, 2006