Behavioural Animation of Virtual Humans: What Kind of Laws and Rules ?

Daniel Thalmann, Jean-Sébastien Monzani
Computer Graphics Lab
EPFL, Switzerland

Abstract

Interactive systems, games, VR and multimedia systems require more and more flexible Virtual Humans with individualities. Behavioral animation seems to be the best way to develop this kind of applications, but there is still a major problem to select the right laws and rules to implement individual but believable behaviors. To create motion laws, there are mainly two approaches: 1) Recording the motion using motion capture systems, then to try to alterate such a motion to create this individuality. This process is tedious and there is no reliable method at this stage. 2) Creating computational models which are controlled by a few parameters. One of the major problem is to find such models and to compose them to create complex motion. Such models can be created for walking, grasping, but also for groups and crowds.

1 Introduction

Virtual humans simulations are becoming each time more popular. Nowadays many systems are available to animate virtual humans. Such systems encompass several different domains as: autonomous agents in virtual environments, human factors analysis, training, education, virtual prototyping, simulation-based design, and entertainment. Virtual humans (see Fig.1) are commonly used nowadays in the entertainment industry, and most specifically in movies and video games. If the quality of pictures has been dramatically improved during the last years, the animation is still a major bottleneck in production. For movies, one can afford to spend months in order to produce a realistic animation for each character, but for real-time applications (and this is particularly true in video games) it is still very difficult to handle the behavior of virtual agents, especially when we try to make them autonomous. In Behavioral Animation, virtual humans acquire the capabilities of perceiving their environment and are able to react and make decisions, depending on this input (it is important to note that agents need to be situated in a common environment: otherwise, no interaction is possible). The question that we are facing is: how to populate virtual environments with virtual humans so that they can behave autonomously. Autonomy will be judged as their capabilities to:

- react to changes in the environment (including other agents)
- reason and make decisions by themselves, based on acquired information or internal stimuli.

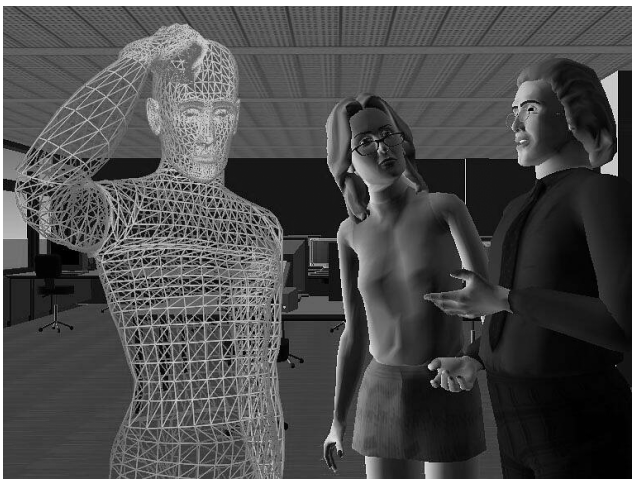

Figure 1. Virtual Humans

2 Behavior

It is indeed not easy to define the notion of behaviour: quoting the Merriam-Webster dictionary, it can be seen as a: the manner of conducting oneself, b: anything that an organism does involving action and response to stimulation, c: the response of an individual, group, or species to its environment. Starting with a system capable of displaying and animating virtual creatures (and especially humans), one can see as a "behaviour" some very simple actions like "turning head to the left" to very general goals such as "go to the closest bank in the city and withdraw enough money to buy something to eat". In this paper, we will generally use terms such as actions or gesture to refer to the most simple behaviours that an agent is able to perform and employ behaviour for more abstract capabilities, such as executing a sequence of actions. Combining actions altogether is indeed a behaviour, even if it also involves some geometric knowledge: for instance, it is possible to walk while taking a book in our hands, while it is impossible to sit and walk at the same time, simply because these two actions are controlling the same body elements. We can see in Figure 2 a typical behavioral engine.

Proceedings of the Computer Animation 2002 (CA 2002)
1087-4844/02 $17.00 © 2002 IEEE

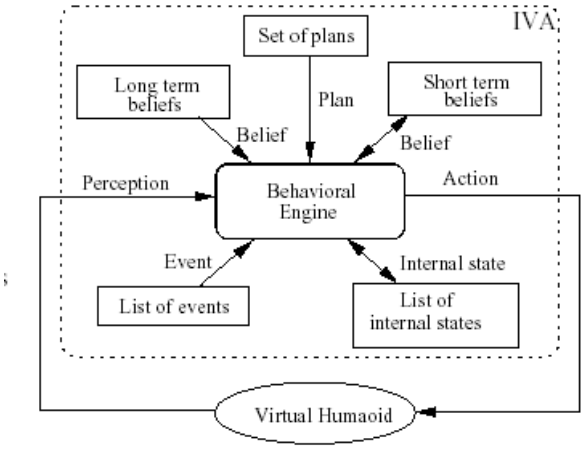

Figure 2. A typical behavioral engine

In the context of Virtual Humans, a Motion Control Method (MCM) specifies how the Virtual Human is animated and may be characterized according to the type of information it privileged in animating this Virtual Human. For example, in a keyframe system for an articulated body, the privileged information to be The problem is basically to be able to generate variety among a finite set of motion requests and then to apply it to either an individual or a member of a crowd. A single autonomous agent and a member of the crowd present the same kind of 'individuality'. The only difference is at the level of the modules that control the main set of actions. With this formulation, one can also see that the personality of an agent (i.e. the set of noisy actions) can be preserved whenever it is in a crowd, alone. Figure 3 shows Virtual Humans in a city park.

Figure 3. Virtual Humans in a city park

The problem is basically to be able to generate variety among a finite set of motion requests and then to apply it to either an individual or a member of a crowd. A single autonomous agent and a member of the crowd present the same kind of 'individuality'. The only difference is at the level of the modules that control the main set of actions. With this formulation, one can also see that the personality of an agent (i.e. the set of noisy actions) can be preserved whenever it is in a crowd, alone.

To create this flexible Virtual Humans with individualities, there are mainly two approaches:

- Recording the motion using motion capture systems (magnetic or optical), then to try to alterate such a motion to create this individuality. This process is tedious and there is no reliable method at this stage.
- Creating computational models which are controlled by a few parameters. One of the major problem is to find such models and to compose them to create complex motion. Such models can be created for walking, running, grasping, but also for interaction, groups, and crowds.

3 Motion capture and retargeting

3.1 Introduction

The first approach consists in recording the motion (Fig. 4) using motion capture systems (magnetic or optical), then to try to alterate such a motion to create this individuality.

.

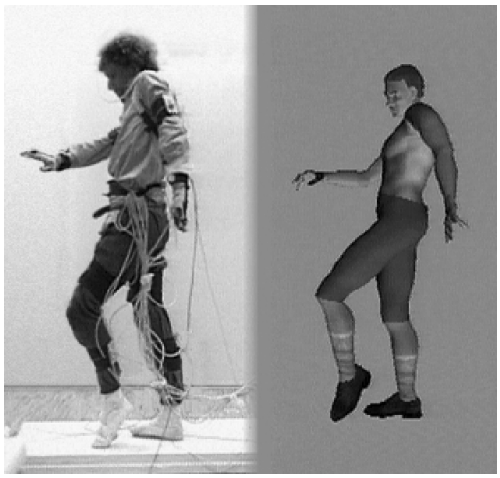

Figure 4. Motion capture

This process is tedious and there is no reliable method at this stage. Even if it is fairly easy to correct one posture by modifying its angular parameters (with an Inverse Kinematics engine, for instance), it becomes a difficult task

to perform this over the whole motion sequence while ensuring that some spatial constraints are respected over a certain time range, and that no discontinuities arise. When one tries to adapt a captured motion to a different character, the constraints are usually violated, leading to problems such as the feet going into the ground or a hand unable to reach an object that the character should grab. The problem of adaptation and adjustment is usually referred to as the Motion Retargeting ProblemWitkin and Popovic [1] proposed a technique for editing motions, by modifying the motion curves through warping functions and produced some of the first interesting results. In a more recent paper [2], they have extended their method to handle physical elements, such as mass and gravity, and also described how to use characters with different numbers of degrees of freedom. Their algorithm is based on the reduction of the character to an abstract character which is much simpler and only contains the degrees of freedom that are useful for a particular animation. The edition and modification are then computed on this simplified character and mapped again onto the end user skeleton. Bruderlin and Williams [3] have described some basic facilities to change the animation, by modifying the motion parameter curves. The user can define a particular posture at time t, and the system is then responsible for smoothly blending the motion around t . They also introduced the notion of motion displacement map, which is an offset added to each motion curve. The Motion Retargeting Problem term was brought up by Michael Gleicher [4] . He designed a space-time constraints solver, into which every constraint is added, leading to a big optimisation problem. He mainly focused on optimising his solver, to avoid enormous computation time, and achieved very good results. Bindiganavale and Badler [5] also addressed the motion retargeting problem, introducing new elements: using the zero-crossing of the second derivative to detect significant changes in the motion, visual attention tracking (and the way to handle the gaze direction) and applying Inverse Kinematics to enforce constraints, by defining six sub-chains (the two arms and legs, the spine and the neck). Finally, Lee and Shin [6] used in their system a coarse-to-fine hierarchy of B-splines to interpolate the solutions computed by their Inverse Kinematics solver. They also reduced the complexity of the IK problem by analytically handling the degrees of freedom for the four human limbs

Lim and Thalmann [7] have addressed an issue of solving customers' problems when applying evolutionary computation. Rather than the seemingly more impressive approach of wow-it-all-evolved- from-nothing, tinkering with existing models can be a more pragmatic approach in doing so. Using interactive evolution, they experimentally validate this point on setting parameters of a human walk model for computer animation while previous applications are mostly about evolving motion controllers of far simpler creatures from scratch. Figure 5 shows an example of such application of evolutionary computation.

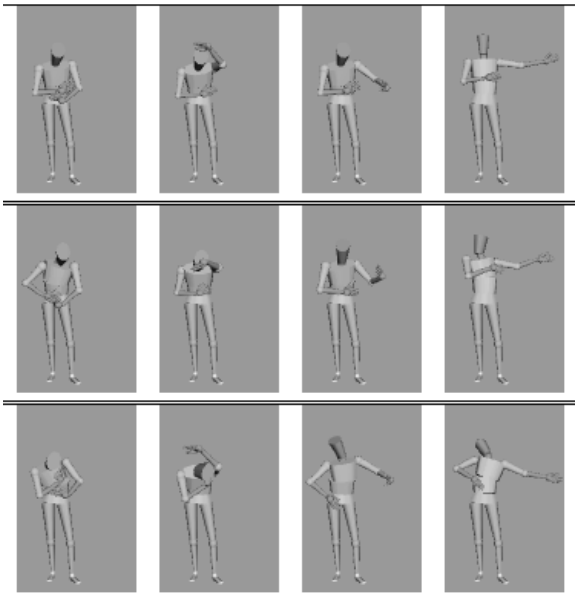

Figure 5. Evolutionary Computation: the original motion of the first row has evolved into the motion in rows 2 and 3

3.2 Using an intermediate skeleton

Given a captured motion associated to its Performer Skeleton, Monzani et al. [8] decompose the problem of retargeting the motion to the *End User Skeleton* into two steps

- First, computing the Intermediate Skeleton matrices by orienting the Intermediate Skeleton bones to reflect the Performer Skeleton posture *(Motion Converte*r).
- Second, setting the End User Skeleton matrices to the local values of the corresponding Intermediate Skeleton matrices.

The first task is to convert the motion from one hierarchy to a completely different one. The Intermediate Skeleton model is introduced to solve this, implying three more subtasks: manually set at the beginning the correspondences between the two hierarchies, create the Intermediate Skeleton and convert the movement. It is then possible to correct the resulting motion and make it enforce Cartesian constraints by using Inverse Kinematics. When considering motion conversion between different skeletons, one quickly notices that it is very difficult to directly map the Performer Skeleton values onto the End User Skeleton, due to their different proportions, hierarchies and axis systems. This raised the idea of having an Intermediate Skeleton: depending on the Performer Skeleton posture, its bones are

reoriented to match the same directions. We have then an easy mapping of the Intermediate Skeleton values onto the End User Skeleton. The first step is to compute the Intermediate Skeleton (Anatomic Binding module). During the animation, motion conversion takes two passes (sse Fig.6), through the Motion Converter and the Motion Composer (which has a graphical user interface).

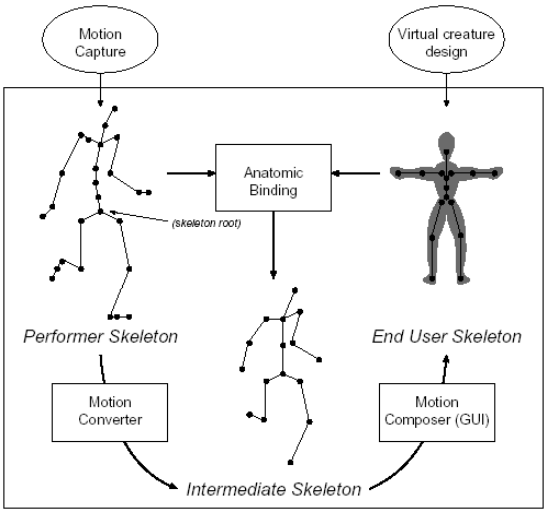

Figure 6. Use of an intermediate skeleton for motion retargeting

An example is shown in Figure 7. The Performer Skeleton (on the right) is going to release an object at the location specified by the ball, and the motion is retargeted onto the End User Skeleton (on the left), by constraining its right hand to also reach another location. When we first set the starting time for the constraint, we notice that at frame 126, the Performer Skeleton hand is very close to the ball, while the End User Skeleton right hand is too far from it.

4 Creating Compositional models

The second approach consists in creating computational models which are controlled by a few parameters. Motion synthesis relies on numerical models which give the body posture at a specific time. It is well suited for physically-correct simulations, and especially for dynamics (like for ball rebounds in a tennis game), but usually fails to parameterise complex human motions. However, it can produce good results in some specific cases, like synthesis of walk [9] or Perlin's work [10].One of the major problem is to find such models and to compose them to create complex motion. Such models can be created for walking or grasping objects, but also for groups and crowds.

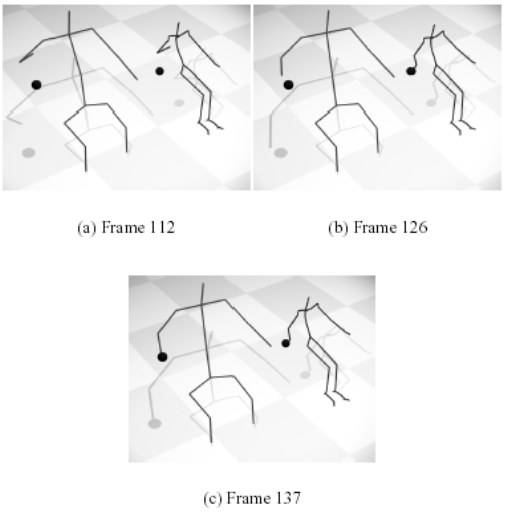

Figure 7. An example of motion retargeting

4.1 Walking

Walking has global and specific characteristics. From a global point of view, every human-walking has comparable joint angle variations. However, at a close-up, we notice that individual walk characteristics are overlaid to the global walking gait. Based on the walking engine described in [9][11], walking is a specialized action in the animation framework where the joint angle variations are synthesized by a set of periodic motions which we briefly mention here:

- sinus functions with varying amplitudes and frequencies for the humanoid's global translations (vertical, lateral and frontal) and the humanoid's pelvic motions (forward/backward, left/right and torsion)
- periodic functions based on control points and interpolating Hermite splines. They are applied to the hip flexion, knee flexion, ankle flexion, chest torsion, shoulder flexion and elbow flexion.

The parameters of the joint angle functions can be modified in a configuration file in order to generate personalized walking gaits, ranging from tired to energetic, sad to happy, smart to silly. The algorithm also integrates an automatic speed tuning mechanism which prevents sliding on the supporting surface. Many high level parameters can be adjusted dynamically, such as linear and angular velocity, foot step locations and the global walk trajectory. The walk engine has been augmented by a specialized action interface and its full capacity is therefore available within the animation framework. The specialized action directly exports most common high level parameter adjustment functions. For fine-tuning, it is still possible to explicitly access the underlying motion generator. With a walking

engine integrated as a specialized action, a walking and phoning human is easily done, simply by performing the walk together with a 'phone'-keyframe for example. In Figure 8, we show some of the parameterized gaits achieved through the specialized action interface.

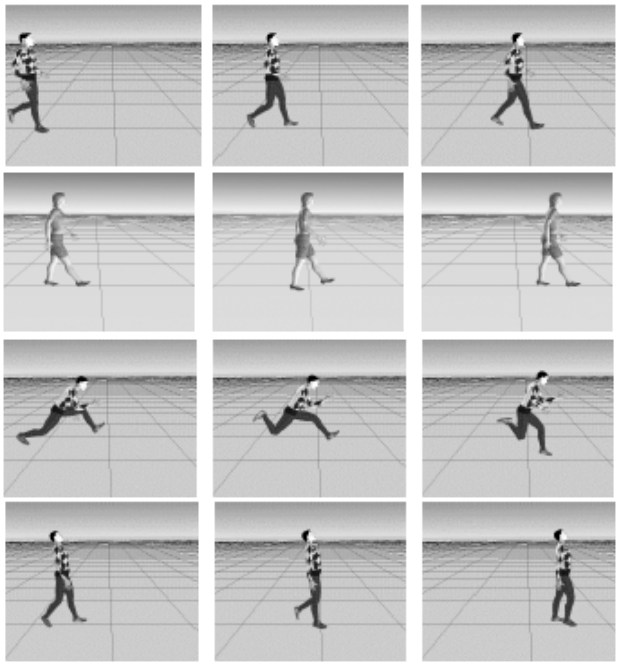

Figure 8. Individualized walking

4.2 Grasping

For grasping object, it is common to consider three steps [12]:

- An *heuristic grasping decision* is based on a grasp taxonomy, Mas and Thalmann [13] proposed a completely automatic grasping system for synthetic actors. In particular, the system can decide to use a pinch when the object is too small to be grasped by more than two fingers or to use a two-handed grasp when the object is too large.
- *Inverse kinematics* is used to find the final arm posture
- *Spherical Multi-sensors* are attached to the articulated hand. Multi-sensor hand. They have both touch and length sensor properties, and have been found very efficient for synthetic actor grasping problem. A sensor is activated for any collision with other objects or sensors.

In case of large objects, such as furniture, grasping simultaneously involves two or more persons. Therefore, we focused on a multi-agent grasp action for encumbering objects. As the object's weight and geometry is distributed over several hand support points of different agents, the heuristic motion planning schemes have to be different than the ones for an object grasp performed by a single individual. For example, a large object might be grasped frontally by the first agent and from behind by the second agent (see Fig. 9).

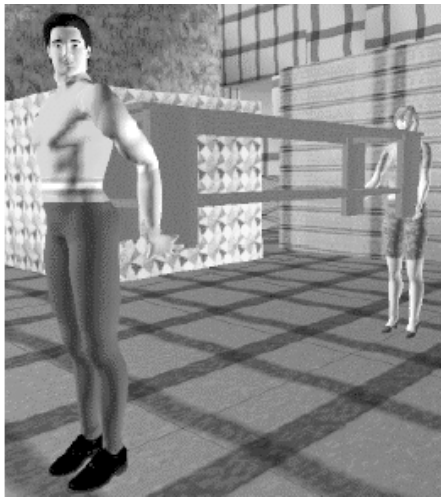

Figure 9. Multi-agent carrying

The humanoid is the active agent, the balloon the passive agent. We can reverse the role of active and passive agent, e.g. the balloon can be active and the human passive (Fig. 10).

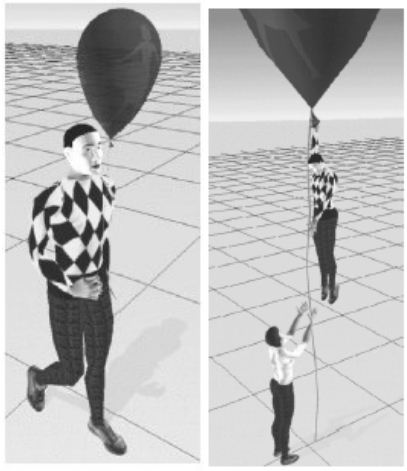

Figure 10. Is the human carrying the balloon or is the balloon lifting the human into the air?

The choice of the active and passive agents depends on which agent is supposed to control the other one – is the human carrying the balloon or is the balloon lifting the human into the air? By extension, any agent can be active and passive at the same time, e.g. a box attaches a balloon and is attached to a humanoid.

5 Crowds and groups

Animating crowds [14] is challenging both in character animation and a virtual city modeling. Though different textures and colors may be used, the similarity of the virtual people would be soon detected by even non-experts, say, "everybody walks the same in this virtual city!" . It is, hence, useful to have a fast and intuitive way of generating motions with different personalities depending on gender, age, emotions, etc., from an example motion, say, a genuine walking motion. The problem is basically to be able to generate variety among a finite set of motion requests and then to apply it to either an individual or a member of a crowd. It also needs very good tools to tune the motion [15].

Reynolds [16] described distributed behavioral model for simulating aggregate motion of a flock of birds. The flock is simulated as a particle system, with the simulated birds (called boids) being the particles. Each boid is implemented as an independent actor that navigates according to its local perception of the dynamic environment, the laws of simulated physics, and a set of behaviors where the boids try to avoid collisions with one another and with other objects in their environment, match velocities with nearby flock mates and move towards center of the flock .The aggregate motion of the simulated flock is the result of the dense interaction of these relatively simple behaviors of the individual simulated birds. Bouvier [17],[18] used combination of particle systems and transition networks to model human crowds in visualization of urban spaces. Lower level enabled people to avoid obstacles using attraction and repulsion forces analogous to physical electric forces. Higher level behavior is modeled by transition networks with transitions depending on timing, visiting of certain points, changes of local densities and global events. Brogan and Hodgins [19] simulated group behavior for systems with significant dynamics. They presented algorithm for controlling the movements of creatures traveling as a group. Algorithm has a two steps: first a perception model determines the creatures and obstacles visible to each individual and then a placement algorithm determines the desired position for each individual given the locations and velocities of perceived creatures and obstacles. Simulated systems included groups of legged robots, bicycle riders and point-mass systems.

Musse and Thalmann's [14] proposed solution addresses two main issues: i) crowd structure and ii) crowd behavior. Considering crowd structure, our approach deals with a hierarchy composed of crowd, groups and agents, where the groups are the most complex structure containing the information to be distributed among the individuals. Concerning crowd behavior, our virtual agents are endowed with different levels of autonomy. They can either act according to an innate and scripted crowd behavior (programmed behavior), react as a function of triggered events (reactive or autonomous behavior) or be guided by an interactive process during simulation (guided behavior). The term <guided crowds> is introduced to define the groups of virtual agents that can be externally controlled in real time [20]. Figure 11 shows a crowd guided by a leader.

Figure 11. Crowd guided by a leader

The intelligence, memory, intention and perception are focalized in the group structure. Also, each group can obtain one leader. This leader can be chosen randomly by the crowd system, defined by the user or can emerge from the sociological rules. Concerning the crowd control features, The crowd aims at providing autonomous, guided and programmed crowds. Varying degrees of autonomy can be applied depending on the complexity of the problem. Externally controlled groups, <guided groups>, no longer obey their scripted behavior, but act according to the external specification. At a lower level, the individuals have a repertoire of basic behaviors that we call innate behaviors. An innate behavior is defined as an "inborn" way to behave. Examples of individual innate behaviors are goal seeking behavior, the ability to follow scripted or guided events/reactions, the way trajectories are processed and collision avoided. While the innate behaviors are included in the model, the specification of scripted behaviors is done by means of a script language. The groups of virtual agents whom we call <programmed groups> apply the scripted behaviors and do not need user intervention during simulation. Using the script language, the user can directly specify the crowd or group behaviors. In the first case, the system automatically distributes the crowd behaviors among the existing groups. Events and reactions have been used to represent behavioral rules. This reactive character of the simulation can be programmed in the script language (scripted control) or directly given by an external controller. We call the groups of virtual agents who apply the behavioral rules <autonomous groups>.

The train station simulation (Figure 12) includes many different actions and places, where several people are present and doing different things. Possible actions include "buying a ticket", "going to shop", "meeting someone", "waiting for someone", "making a telephone call", "checking the timetable", etc. This simulation uses external control (RBBS [21][22]) to guide some crowd behaviors in real time.

Figure 12. Train station simulation.

For emergent crowds, Ulicny and Thalmann [23] proposed a behavior model based on combination of rules [22],[24] and finite state machines [25],[26] for controlling agent's behavior using layered approach. First layer deals with the selection of higher-level complex behavior appropriate to agent's situation, second layer implements these behaviors using low-level actions provided by the virtual human [27]. At the higher level, rules select complex behaviors (such as flee) according to agent's state (constituted by attributes) and the state of the virtual environment (conveyed by events). In rules, it is specified for whom (e.g. particular agent, or agents in particular group) and when the rule is applicable (e.g. at defined time, after receiving event or when some attribute reached specified value), and what is the consequence of rule firing (e.g. change of agent's high-level behavior or attribute). Example of such rule is:

```
FOR ALL
        WHEN EVENT = in_danger_area AND
        ATTRIBUTE fear > 50%
THEN BEHAVIOR FLEE
```

At the lower level, complex behaviors are implemented by hierarchical finite state machines. Each behavior is realized by one FSM which drives selection of the low-level actions for the virtual human (like move to location, play short animation sequence), manages connections with the environment (like path queries, or event sending) and also can call other FSMs to delegate subtasks such as path following

6 Synthetic vision and memory

Let's now consider the simulation of a referee during a tennis match. He has to decide if the ball is out or in. One solution is to calculate the intersection between the impact point of the ball and the court lines. Such an analytical calculation will lead to the decision that the ball is out for 0.01 millimeters. Ridiculous, nobody in reality could take such an objective decision, this is not believable. The decision should be based on the evaluation of the visual aspect of the scene as perceived by the referee.

In a more general context, it is tempting to simulate perception by directly retrieving the location of each perceived object straight from the environment. This is of course the fastest solution (and has been extensively used in video-games until the mid-nineties) but no one can ever pretend that it is realistic at all (although it can be useful, as we will see later on). Consequently, various ways of simulating visual perception have been proposed, depending on whether geometric or semantic information (or both) are considered. Renault et al. introduced first the concept of synthetic vision [28] then extended by Noser et al..[29]. Tu and Terzopoulos [30] implemented a realistic simulation of artificial fishes. Other authors [31] [32] [33] also provided synthetic vision approaches. In the next section, we are going to compare now rendering-based vision, geometric vision and database access.

6.1 Rendering-based vision

Rendering-based vision from Noser and Renault et al. [29] is achieved by rendering of-screen the scene as viewed by the agent. During the process, each individual object in the scene is assigned a different colour, so that once the 2D image has been computed, objects can still be identified: it is then easy to know which object is in sight by maintaining a table of correspondences between colours and objects' IDs. Furthermore, highly detailed depth information is retrieved from the view z-buffer, giving a precise location for each object. An other application of synthetic vision is real-time collision avoidance for multiple agents: in this case, each agent is perceiving the others, and dynamically creates local goals so that it avoids others while trying to reach its original global goal.

Rendering-based vision is the most elegant method, because it is the more realistic simulation of vision and addresses correctly vision issues such as occlusion for instance. However, rendering the whole scene for each agent is very costly and for real-time applications, one tend to favour geometric vision.

One problem is how to decide that an object is in the field of view of the Virtual Human and that he/she can identify it. We can imagine for example that the Virtual Human's wife is in front of the VH but hidden by a wardrobe and on the computed 2D image contains only one pixel for the wife, can he recognize his wife based on such a detail ?

6.2 Geometric vision

Bordeux [34] has proposed a perception pipeline architecture (see Fig.13) into which filters can be combined to extract the required information. The perception filter represents the basic entity of the perception mechanism. Such a filter receives a perceptible entity from the scene as

input, extracts specific information about it, and finally decides to let it pass through or not.

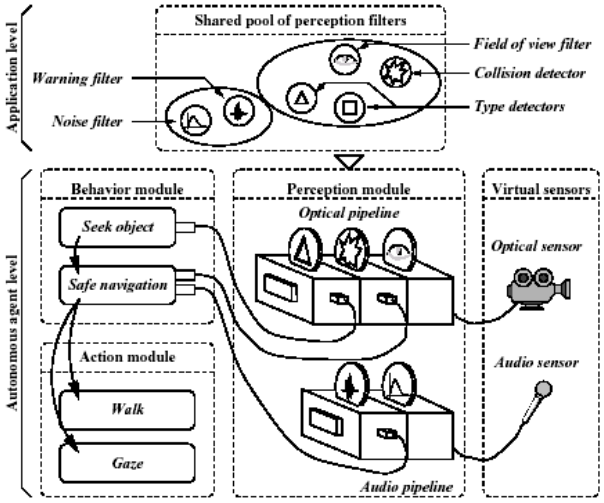

Figure 13. Organization for Geometric vision

The criteria used in the decision process depends on the perception requirements. For virtual objects, they usually involve considerations about the distance and the relative direction of the object, but can also be based on shape, size, colour, or generic semantic aspects, and more generally on whatever the agent might need to distinguish objects. Filters are built with an object oriented approach: the very basic filter for virtual objects only considers the distance to the object, and its descendants refine further the selection.

Actually, the structure transmitted to a filter contains, along with the object to perceive, a reference to the agent itself and previously computed data about the object. The filter can extend the structure with the results of its own computation, for example the relative position and speed of the object, a probable time to impact or the angular extension of the object from the agent s point of view. Since a perception filter does not store data concerning the objects that passed through it, it is fully reentrant and can be used by several agents at the same time. This allows the creation of a common pool of filters at the application, each agent then referencing the filters it needs, thus avoiding useless duplication.

As an example of filters, Bordeux has implemented a basic range filter which selects objects in a given range around the agent. The field of view filter simulates an agent field of view with a given angular aperture. The collision filter detects potential impacts with other objects in the agent neighborhood and estimates, if needed, the time to impact, the objects relative speed and a local area to escape from. This has been used again in a safe-navigation behaviour which dynamically computes a collision-free path through the world. It is even possible to specify how long an object shall stay in the list after it was perceived, in order to simulate short-term memory.

However, the major problem with Geometric vision is to find the proper formulas when intersecting volumes (for instance, intersecting the view frustum of the agent with a volume in the scene). One can use bounding boxes to reduce the computation time, but it will always be less accurate than Synthetic vision. Nevertheless, it can be sufficient for many applications and, as opposed to rendering-based vision, the computation time can be adjusted precisely by refining the bounding volumes of objects.

6.3 Database access

Data access makes maximum use of the scene data available in the application, which can be distributed in several modules. For instance, the objects position, dimensions and shape are maintained by the rendering engine whereas semantic data about objects can be maintained by a completely separate part of the application. Due to scalability constraints as well as plausibility considerations, the agents generally restrain their perception to a local area around them instead of the whole scene. This method is generally chosen when the number of agents is high, especially with crowds, like in Reynolds's [16] flocks of birds and schools of fishes. In Musse's [20] crowd simulation, human agents directly know the position of their neighbours and compute coherent collision avoidance trajectory. As said before, the main problem with the method is the lack of realism, which can only be alleviated by using one of the other methods.

These various approaches to visual perception have their advantages and disadvantages dependent essentially of the complexity and the context of the scenes. But, finally no approach can solve common problematics as the following one: What makes a little girl to be lost in a crowd ? The child will be lost if she just does not know where is her family. Now imagine a virtual crowd where each individual is indexed. It will be extremely easy fo find where is the girl (index 345) and the parents (index 748). At this stage, we could just activate a function making the girl walking towards his parents. This is completely unrealistic from a behavioural point of view.

6.4 Memory

Noser et al. [29] made a few years ago a character trying to find the exit from a maze. To simulate the memory process, they used an octree structure to store the information see by the character. The results were that the second time, it was straightforward for the character to find the exit. Again, this is not so convincing as never somebody could remember all the paths inside a maze. This kind of memory can then easily be linked to the synthetic vision: the 2D rendering and the corresponding z-buffer data are

combined in order to determine whether the corresponding voxel of the scene is occupied by an object or not. By navigating through the environment, the agent will progressively construct a voxel-based representation of it. Of course, a rough implementation of this method would suffer from dramatic memory cost, because of the high volume required to store all voxels. Noser proposed to use octrees instead which successfully reduces the amount of data. Once enough information has been gathered through exploration, the agent is then able to locate things and find its way.

Peters and O'Sullivan [33] propose a system of memory based on what is referred to a "stage theory" by Atkinson and Shiffrin [35]. They propose a model where information is processed and stored in 3 stages: sensory memory, short-term memory, and long-term memory.

Although these approaches are quite interesting, they do not solve the following simple problematics. Imagine now a Virtual Human inside a room containing 100 different objects. Which objects can we consider as memorized by the Virtual Human ? Can we decide that when an object is seen by the actor, it should be stored in his memory. To answer this question, we have just to consider the popular family game consisting in showing 20 objects during 2 minutes to people and asking them to list the objects. Generally nobody is able to list the 20 objects. Now, how to model this inability to remember all objects ?

7 Conclusion

In order to develop truly interactive multimedia systems with Virtual Humans, games, and interactive movies, we need a flexible way of animating these Virtual Humans. Altering motion obtained from a motion capture system is not the best solution. Only computational models can offer this flexibility unless powerful motion retargeting methods are developed, but in this case they will look similar to computational models.

8 Acknowledgments

The authors would like to thank all people who have contributed to these projects especially Luc Emering, Soraia Musse, Ik Soo Lim, Branislav Ulicny, and Mireille Clavien. Research has been partly funded by the Swiss National Foundation for Research and the Federal Office for Education and Science.

9 References

1 A.Witkin, Z.Popovic. Motion warping. Proceedings of SIGGRAPH 95, pages 105–108, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

2 Z.Popovic, A.Witkin. Physically based motion transformation. Proceedings of SIGGRAPH 99, pages 11–20, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

3 A.Bruderlin, L.Williams. Motion sig-nal processing. In Robert Cook, editor, SIGGRAPH 95 Conference Proceedings, Annual Conference Se-ries, pages 97–104. ACM SIGGRAPH, Addison Wes-ley, August 1995. held in Los Angeles, California, 06- 11 August 1995.

4 M.Gleicher. Retargeting motion to new characters. In Michael Cohen, editor, SIGGRAPH 98 Con-ference Proceedings, Annual Conference Series, pages 33–42. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

5 R.Bindiganavale, N. I. Badler. Motion abstraction and mapping with spatial constraints. In N. Magnenat-Thalmann and D. Thalmann, editors, Modeling and Motion Capture Techniques for Virtual Environments, Lecture Notes in Artificial Intelligence, pages 70–82. Springer, November 1998. held in Geneva, Switzerland, November 1998.

6 L.Jehee, S.Y.Shin. A hierarchical approach Proceedings of SIGGRAPH 99, pages 39–48, Au-gust 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.

7 I.S.Lim, D.Thalmann, Solve Customers' Problems: Interactive Evolution for Tinkering with Computer Animation, Proc. 2000 ACM Symposium on Applied Computing (SAC2000), pp. 404-407.

8 J.-S. Monzani, P. Baerlocher, R. Boulic, D. Thalmann, Using an Intermediate Skeleton and Inverse Kinematics for Motion Retargeting, Proc. Eurographics 2000, pp.11-19

9 R.Boulic, N.Magnenat-Thalmann, D.Thalmann, A Global Human Walking Model with Real-time Kinematics Personification ,The Visual Computer, Vol.6, No6, December 1990, pp.344-358.

10 Ken Perlin and Athomas Goldberg. Improv: A system for scripting interactive actors in virtual worlds. Proceedings of SIGGRAPH 96, pages 205–216

11 R.Boulic, T.Capin, Z.Huang, L.Moccozet, T.Molet, P.Kalra, B.Lintermann, N.Magnenat-Thalmann, I.Pandzic, K.Saar, A.Schmitt, J.Shen, D.Thalmann, The HUMANOID Environment for Interactive Animation of Multiple Deformable Human Characters, Proc. Eurographics `95, Maastricht, August 1995, pp.337-348.

12 Z.Huang, R.Boulic, N.Magnenat-Thalmann, D.Thalmann, A Multi-sensor Approach for Grasping and 3D Interaction, Proc. Computer Graphics International `95, Leeds, Academic Press, pp.235-254.

13 R.Mas, D.Thalmann, A Hand Control and Automatic Grasping System for Synthetic Actors, Proc. Eurographics '94, Oslo

14 S.R. Musse, D.Thalmann, A Behavioral Model for Real-Time Simulation of Virtual Human Crowds, IEEE Transactions on Visualization and Computer Graphics, Vol.7, No2, 2001, pp.152-164.

15 L.Emering, R.Boulic, T.Molet, D.Thalmann, Versatile Tuning ofHumanoid Agent Activity, Computer Graphics Forum

16 Reynolds, C. W., "Flocks, Herds, and Schools: A Distributed Behavioral Model", Computer Graphics, 21(4) (SIGGRAPH '87 Conference Proceedings) pp. 25-34, 1987.

17 Bouvier, E., Guilloteau, P., "Crowd Simulation in Immersive Space Management", Proc. Eurographics Workshop on

Virtual Environments and Scientific Visualization '96, pp. 104-110, Springer-Verlag, 1996.
18 Bouvier, E., Cohen, E., Najman, L., "From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation", Journal of Electrical Imaging, 6(1), pp.94-107, January, 1997.
19 Hodgins, J., Brogan, D., "Robot Herds: Group Behaviors for Systems with Significant Dynamics", Proc. Artificial Life IV, pp.319-324, 1994.
20 Musse, S.R., Babski, C., Capin, T. and Thalmann, D. Crowd, Modelling in Collaborative Virtual Environments. ACM VRST '98, Taiwan
21 Farenc, N., Musse, S.R, Schweiss, E., Kallmann, M., Aune, O., Boulic, R and Thalmann, D. "A Paradigm for Controlling Virtual Humans in Urban Environment Simulations". Special Issue on Intelligent Virtual Environments, 1999.
22 Schweiss, E., Musse, S.R.; Garat, F. "An Architecture to Guide Crowds based on rule-based systems". Autonomous Agents '99, Seattle, Washington, USA, 1999.
23 Ulicny, B., Thalmann, D., "Crowd simulation for interactive virtual environments and VR training systems", Proc. Eurographics Workshop on Animation and Simulation'01, Springer-Verlag, 2001.
24 Rosenbloom, P.S., Laird, J.E., Newell, A., "The Soar papers: Research on Artificial Intelligence", MIT Press, 1993.
25 Cremer, J., Kearney, J., and Papelis, Y., "HCSM: Framework for Behavior and Scenario Control in Virtual Environments", ACM Transactions on Modeling and Computer Simulation, 5(3):242-267, 1995.
26 Motivate product information, Motion Factory, http://www.motion-factory.com
27 Boulic, R., Becheiraz, P., Emering, L., and Thalmann, D., "Integration of Motion Control Techniques for Virtual Human and Avatar Real-Time Animation", Proc. VRST '97, pp. 111-118, ACM Press, 1997.
28 Renault O.Renault, N. Magnenat-Thalmann, D. Thalmann, A Vision-based Approach to Behavioural Animation, Journal of Visualization and Computer Animation, Vol.1, No1, 1990, pp.18-21.
29 Noser H. Noser, O. Renault, D. Thalmann, N. Magnenat Thalmann, Navigation for Digital Actors based on Synthetic Vision, Memory and Learning, Computers and Graphics, Pergamon Press, Vol.19, No1, 1995, pp.7-19.
30 X.Tu, D.Terzopoulos, Artificial Fishes, Physics, Locomotion, Perception, Behaviour, Proc. SIGGRAPH '94, pp.43-50.
31 J.Kuffner, J.C.Latombe, Fast Synthetic Vision, Memory, and Learning Models for Virtual Humans, Proc. Computer Animation 1999, IEEE CS Press, pp.118-127.
32 B.M.Blumberg, T.A,Galyean, Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments, Proc. SIGGRAPH 95, 1995, pp.47-54.
33 C.Peters, C.O'Sullivan, A Memory Model for Autonomous Virtual Humans, Proc. Third Irish Eurographics Workshop on Computer Graphics, Dublin, pp. 21-26.
34 C.Bordeux, R. Boulic, D.Thalmann, An Efficient and Flexible Perception Pipeline for Autonomous Agents, Proc. Eurographics '99, Milano, Italy, pp.23-30.
35 R.Atkinson, R Shiffrin, Human Memory : a Proposed System and its Control Processes, in: K.Spence and J.Spence, the Psychology of Learning and Motivation: Advances in Research and Theory, Vol.2, NY, Academic Press, 1968.