Noname manuscript No.
(will be inserted by the editor)

Generating Steering Behaviors for Virtual Humanoids using BVP Control

Fábio Dapper, Edson Prestes, Luciana P. Nedel

Instituto de Informática – Universidade Federal do Rio Grande do Sul
Av. Bento Gonçalves, 9500 – Campus do Vale - Bloco IV
Porto Alegre – RS – Brasil 91501-970
e-mail: {fdapper,prestes,nedel}@inf.ufrgs.br

The date of receipt and acceptance will be inserted by the editor

Abstract One of the main challenges on animating embodied autonomous characters in real-time applications is the ability to generate believable behaviors, more precisely, actors capable of moving in a natural and improvisational manner. In this paper we propose an elegant and low cost solution based upon boundary value problems (BVP) to control steering behaviors of characters. We use a field potential formalism that allows synthetic actors to move negotiating space, avoiding collisions, and attaining goals, while producing very individual paths. The individuality of each character can be set by changing its inner field parameters leading to a broad range of possible behaviors. To illustrate the technique potentialities, some results exploring situations as steering behavior in corridors with collision avoidance and competition for a goal, and searching for objects in unknown environments are presented and discussed.

Key words Humanoid Simulation Path Planning Steering behavior Harmonic Functions Boundary Value Problems

1 Introduction

In interactive applications such as games and virtual reality experiences, autonomous agents (also called *non-player characters*) are characters with the ability of playing a role into the environment with life-like and improvisational behavior [14]. Suitable skills for these characters (often simulating human beings) include: a realistic appearance, the ability to produce natural movements, and the aptitude to reason and act in an unforeseeable way. However, the high performance required for the algorithms used on real-time graphics applications frequently compel developers to look for better methods to generate more natural and unexpected simple movements. In this way, it is possible to improve the applications behavior quality avoiding the high cost frequently imposed by the use of AI methods.

The simulation of virtual humans moving into a synthetic world involves the environment specification, the definition of the agent initial position and its goal (target position). By setting these parameters, a path-planning algorithm can be used to find a trajectory to be followed. However, in the real world, if we consider several persons (all in the same initial position) looking for achieving the same target position, each path followed will be unique. Even for the same task, the strategy used for each person to reach his/her goal will depend on his/her physical constitution, personality, mood and reasoning. In this work we propose an algorithm to generate interesting behaviors for humanoids, considering that, from a single path, several behaviors can be explored to drive the agent from one position to another.

Despite *humanoid*, *autonomous agent*, and *behavior* are terms used in many different contexts, in this paper we will limit its use to match our goals. For sake of simplicity, we consider *humanoids* as a kind of embodied *autonomous agent* with reactive behaviors (driven by stimulus), represented by a computational model, and capable of producing physical manifestations in a virtual world. The term *behavior* will be used mainly as a synonymous of *animation* or *motion behavior* and intend to refer the improvisational and personalized action of a *humanoid*.

In a previous work [4] we proposed a method based on the numeric solution of the boundary value problem (BVP) to control pedestrians. We showed that a single principle can be used to generate interesting and complex human-like behaviors while humanoids move to achieve a navigational task. In this paper we propose some improvements for our initial algorithm, varying the motion speed and proposing new possibilities to follow a path. Some very first experiments towards to endow the

humanoids the ability to explore unknown environments are also presented.

The remaining of this paper is structured as follows. Section 2 reviews some related work on path-planning techniques applied to virtual humans simulation. Section 3 describes the fundamentals of the path planning method proposed by us, as well as how we solve the BVP. In Section 4 we detail the strategy used to handle the information about the environment and other agents and in Section 5 how the movement and velocity of the agent are managed. Finally, Section 6 presents our results and Section 7 conclusions and future works.

2 Related Work

Thanks to the researches in robotics, the path-planning problem is almost solved. However, in the computer graphics domain, to find a natural and realistic way to move a character is as important as to find a path between two points. In order to generate realistic results and allow its use in real-time applications, several authors proposed motion planning solutions based on two steps. In general, the first step is dedicated to define a valid path, while the second adapts this path to generate a more realistic movement.

Kuffner [5] proposed a technique with the first step dedicated to path-planning and the second to path-following. The 3D scenario is projected in 2D and the humanoid treated as a disc, reducing the dimension of the planning problem. Metoyer and Hodgings [10] proposed a similar technique also based on two steps. In their method, the characters have a pre-defined path to follows and this path is smoothed and slightly changed to avoid collisions based on force fields.

The development of randomized path-finding algorithms – specially the PRM (Probabilistic Roadmaps) [6] and RTT (Rapidly-exploring Random Tree) [8] – allow the use of large and most complexes configuration spaces, and generating paths most efficiently. Thus, the challenge becomes more the generation of realistic movements than finding a valid path.

Choi *et al.* [2] proposed the use of a library of captured movements associated to PRM to generate realistic movements in a static environment. Despite the fact the path maps should be generated in a pre-processing phase, the results are very realistic. Pettré *et al.* [11] used a PRM to identify a free of collisions path and Bézier curves to generate smooth paths associating it with a motion library. As in the previous works, the motion is also performed on a 2D environment.

Differently, Burgess and Darken [1] proposed a method to obtain very realistic paths through a terrain using properties of fluid simulation to produce human-like movements. The authors consider that a realistic path for a human is the one requiring the smallest amount of effort.

The most part of the works developed since now propose methods based on two separate phases. In next sections we present our own proposal for generating realistic paths based on a single phase. Our assumption is that realistic paths derive from human personal characteristics and internal state, thus varying from one person to another.

3 BVP-Path Planner

Recently, we proposed a framework for controlling virtual humanoids in navigational tasks. It is based on potential fields that do not have local minima [4, 15] generated through the numeric solution of the BVP using Dirichlet boundary conditions and the following equation

$$\nabla^2 \, p(\mathbf{r}) + \epsilon \mathbf{v}.\nabla p(\mathbf{r}) \; = \; 0 \qquad (1)$$

where $\mathbf{v}$ is a bias vector and ϵ is a scalar value.

The allowed values of the parameters ϵ and $\mathbf{v}$ generate an expressive amount of action sequences that virtual humanoids (agents) can take to reach a specific target (goal position). Each action corresponds to a particular displacement that the agent performs at each step. Two sequences are not statically defined for a same pair ϵ and $\mathbf{v}$. They vary according to the information gathered by the agent to allow it to react dynamically against unexpected events (e.g. dynamic obstacles). Satisfactory adjustments of parameters ϵ and $\mathbf{v}$ generate realistic steering behaviors for agents [4].

The core of the Equation 1 is the vector $\mathbf{v}$, so called *behavior vector*, that acts as an external force pulling the agent to its direction always as possible. The parameter ϵ can be understood as the *strength* or *influence* of this vector in the agent behavior. When $\epsilon = 0$, Equation 1 can be reduced to

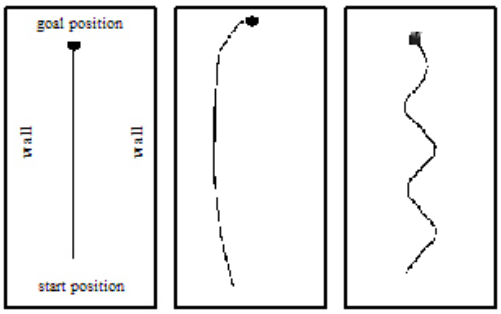
$$\nabla^2 \, p(\mathbf{r}) = 0$$

which is the Laplace's equation and the path planner is called harmonic functions path planner. It has been developed by Connolly and Grupen [3] and one of its features is to lead the agent to a path that minimizes the collision probability.

Figure 1 shows some paths produced using the equation of Laplace and the Equation 1. In Figure 1a, Laplace's equation conducts the agent through a path equidistant to the walls, which is not always adequate to simulate humanoid motion since it looks very stereotyped. In Figure 1b-c, we can observe that adding the term $\epsilon \mathbf{v}.\nabla p(\mathbf{r})$ allows the generation of different kind of paths leading the agent, for instance, closer to the wall.

Our method starts with the discretization of the environment into a fixed homogeneous mesh with identical cells, like an occupancy grid. Each cell (i, j) is associated to a squared region of the real environment and stores a potential value $p_{i,j}$. Dirichlet boundary conditions are

such that, the cells with high probability of having an obstacle are set to 1 (*high potential*) while cells containing the target are set to 0 (*low potential*). The high potential value prevents the agent from running into obstacles whereas the low potential value generates an attraction basin that pulls the agent.



(a) (b) (c)

Fig. 1 Different paths followed by agents using Equation 1: (a) path produced by Laplace's equation, i.e., with $\epsilon = 0$; (b) with $\epsilon = 0.8$ and $\mathbf{v} = (1, 0)$; (c) with $\epsilon = 0.8$ and $\mathbf{v} = (1, \sin(\omega * t))$.

Solving the BVP thus consist in interpolating the potential values on the grid between the obstacles and the target. This can be done using the Gauss-Seidel algorithm which updates the potential cells according to the equation

$$\underbrace{p_{i-1,j} + p_{i+1,j} + p_{i,j-1} + p_{i,j+1} - 4p_{i,j}}_{\nabla^2 p(\mathbf{r})} + \underbrace{\epsilon\left(\frac{(p_{i+1,j} - p_{i-1,j})}{2}v_x + \frac{(p_{i,j+1} - p_{i,j-1})}{2}v_y\right)}_{\epsilon \mathbf{v}.\nabla p(r)} = 0 \quad (2)$$

that leads us directly to the update rule

$$p_{i,j} = \frac{1}{4}(p_{i-1,j} + p_{i+1,j} + p_{i,j-1} + p_{i,j+1}) + \frac{\epsilon}{8}((p_{i+1,j} - p_{i-1,j})v_x + (p_{i,j+1} - p_{i,j-1})v_y) \quad (3)$$

where $\mathbf{v} = (v_x, v_y)$ and $\epsilon \in [-2, +2]$.

ϵ must be in the interval $[-2, +2]$, otherwise, the boundary conditions that assert the agent – repelling obstacles and attracting the target – are violated. Then the method generates oscillatory and unstable behaviors that do not guarantee the agent will reach the target.

The agent uses the gradient descent of this potential to determine the path to follows towards the target position. This method is formally complete, i.e., if there is a path connecting the agent position to the target, it will be found.

4 Environment Management

As explained in last section, our path planning method requires the environment discretization into a regular grid. In this section we present a strategy to implement it by using global environment maps (one for each target) and local maps (one for each agent) to enhance the algorithm performance, allowing the use of our method for real-time applications.

4.1 Environment Global Map

The entire environment is represented by a set of homogeneous meshes $\{m_k\}$, where each mesh m_k is associated to an achievable target o_k and has $L_x \times L_y$ cells, denoted by $\{c^k_{i,j}\}$. Each cell $c^k_{i,j}$ corresponds to a squared region centered in environment coordinates $r = (r_i, r_j)$ and stores a particular potential value $p^k_{i,j}$. Each mesh m_k stores a potential field computed by the harmonic path planner [3] that is used by agents to reach the target o_k.

In order to delimit the navigation space of agents, we consider the environment is surrounded by static obstacles. Global maps are built before the simulation starts.

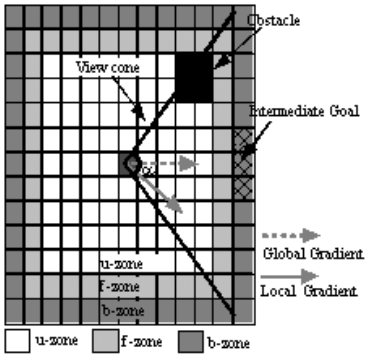


Fig. 2 Agent Local Map. White, light gray and dark gray cells comprise the *update*, *free* and *border zones*, respectively. Red, black and blue cells correspond to the intermediate goal, obstacles and the agent position, respectively.

4.2 Agent Local Map

Each agent a_k has one map am_k that stores the current local information about the environment obtained by its sensors. This map is centered in the current position of the agent and represents a small fraction of the global map. The area associated to each agent map cell is smaller than the area associated to the global map cell. The main reason is that the agent map is used to produce refined motion, hence, the smaller cell size the better the quality of motion; while the global map is used only to assist the long-term agent navigation.

The map am_k has $l_x^k \times l_y^k$ cells, denoted by $\{ac_{i,j}^k\}$ and is divided in three regions: the update zone (*u-zone*); the free zone (*f-zone*) and the border zone (*b-zone*), as shown in Figure 2. In a similar way, each cell corresponds to a squared region centered in environment coordinates $r = (r_i, r_j)$ and stores a particular potential value $ap_{i,j}^k$.

4.3 Updating Local Maps from Global Maps

For each agent a_k, a goal $o_{goal(k)}$, – where the function *goal()* maps the agent number k into its current target number – a particular vector $\mathbf{v}_k$, that controls its behavior, and a ϵ_k that determines the influence of $\mathbf{v}_k$, should be stated. The same goal, $\mathbf{v}$ and ϵ can be designated to several agents. Vector $\mathbf{v}_k$ and ϵ_k can be either static or dynamic. If a variable is dynamic, then the function that controls it must be specified.

To navigate into the environment, an agent a_k uses its sensors to perceive the world and to update its local map with the information about obstacles and other agents. The agent sensor set a view cone with aperture α.

Figure 2 sketches a particular instance of the agent local map. The *u-zone* cells $ac_{i,j}^k$ that are inside the view cone and correspond to obstacles or other agents have their potential value set to 1. In Figure 3, as the agent 1 is inside the *u-zone* of agent 2 local map but out of its view cone, it is not mapped as an obstacle into the local map of agent 2. This procedure assures that dynamic or static obstacles behind the agent do not interfere in its future motion.

For each agent a_k, the global descent gradient on the cell in the global map $m_{goal(k)}$ that contains its current position is calculated. The gradient direction is used to generate an intermediate goal in the border of the local map, setting the potential values to 0 of a couple of *b-zone* cells, while other *b-zone* cells are considered as obstacles, with their potential values set to 1. In Figure 3, each agent calculates its global gradient in order to project an intermediate goal in its own local map. As the agent local map is delimited by obstacles, the agent is pulled towards the intermediate goal using the direction of its local gradient. The intermediate goal helps the agent a_k to reach its target $o_{goal(k)}$ while allowing it to produce a particular motion.

In some cases, the target $o_{goal(k)}$ is inside both the view cone and the *u-zone*, and consequently, local map cells associated are set to 0. The intermediate goal is always projected, even if the target is mapped onto the *u-zone*. Otherwise, the agent can easily get trapped because it would be taking into consideration only the local information about the environment, in a same way as traditional potential fields [7].

F-zone cells are always considered free of obstacles, even when there are obstacles inside. The absence of this zone may close the connection between the current agent cell and the intermediate goal due to the mapping of obstacles in front of the intermediate goal. When this occurs, the agent gets lost because there is no information coming from the intermediate goal to produce a path to reach it. *F-zone* cells handle the situation, always allowing the propagation of the information about the goal to the cells associated to the agent position.

After the sensing and mapping steps, the agent updates the potential value of all the cells of its map using Equation 3 with its pair $\mathbf{v}^k$ and ϵ^k. The local potential is partially relaxed [12] and the agent uses the gradient descent of its position defined by

$$\mathbf{dgrad}^k = \left(\frac{ap_{p_x+1,p_y}^k - ap_{p_x-1,p_y}^k}{2}, \frac{ap_{p_x,p_y+1}^k - ap_{p_x,p_y-1}^k}{2} \right)$$

to determine its displacement. In the local map am_k, $p_x = \lceil l_x^k/2 \rceil$ and $p_y = \lceil l_y^k/2 \rceil$.

5 Updating the Position and Speed of Agents

In our previous work [4], the agent position at time t is computed using the following equation

$$\mathbf{pos}^t = \mathbf{pos}^{t-1} + step \frac{\mathbf{dgrad}}{||\mathbf{dgrad}||} \tag{4}$$

where *step* is a constant that corresponds to the maximum agent displacement[1]. However, during the experiments, we observed that, for several scenarios, this equation failed in producing realistic steering behaviors, as observed in real world. One of the reasons is that the agent changes its direction based solely on the gradient descent of its position. For instance, if the agent local map is small, its reaction time will be very short to treat dynamic obstacles. Then, these obstacles will produce a strong repel force that will change the agent direction abruptly. As we can see in Figure 4, if the agent uses only the gradient descent it will change its direction in nearly 90°.

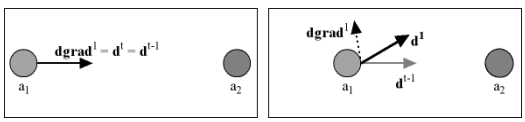


Fig. 4 Agent displacement scheme

We handle this problem by changing Equation 4 into,

$$\mathbf{pos}^t = \mathbf{pos}^{t-1} + step \frac{\mathbf{d}^t}{||\mathbf{d}^t||} \tag{5}$$

with

$$\mathbf{d}^t = \eta\, \mathbf{d}^{t-1} + (1 - \eta)\, \mathbf{dgrad}^t$$

[1] This section presents the equations used by all agents. Therefore, to make the exposion clearer we supress the superscript of the terms that individualize each agent.

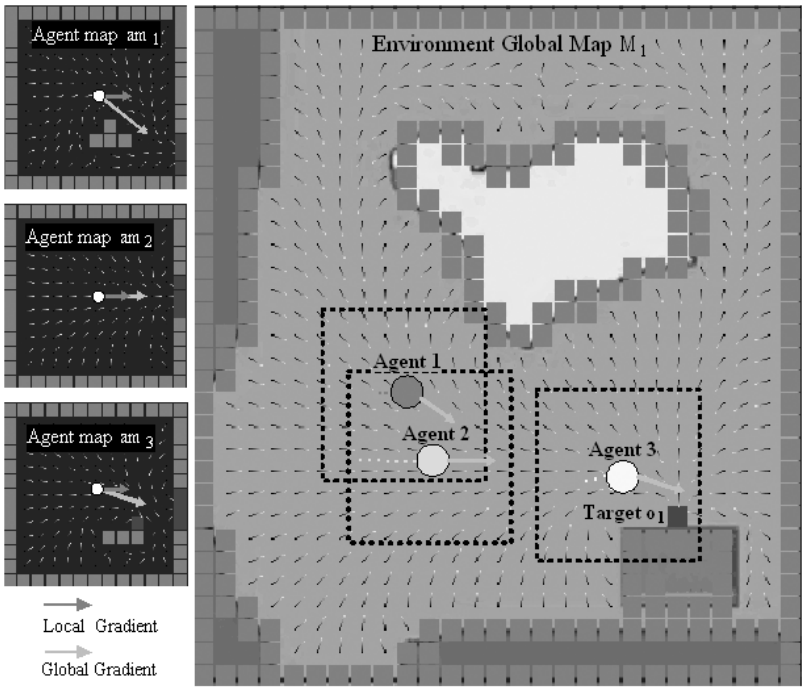


Fig. 3 Agents acting in an environment. Each agent senses the environment, updates its local map and navigates towards the target o_1. Obstacles are represented as (*red squares*) in both global environment map and agent local map; the target o_1 is represented as a blue square in the global environment map and the intermediate goal generated by each agent is represented by a (*blue square*) in its local map.

where $\eta \in [0, 1]$.

If $\eta = 0$, this equation reduces to Equation 4. If $\eta = 0.5$, the previous agent direction ($\mathbf{d^{t-1}}$) and the gradient descent ($\mathbf{dgrad}^t$) influence equally the computation of the new agent direction. Figure 4 shows the vector d^t computed with $\eta = 0.5$. The parameter η can be viewed as an inertial factor that tends to keep constant the agent direction insofar $\eta \to 1$. When $\eta \to 1$, the agent reacts slowly to unexpected events, increasing its hitting probability with obstacles.

Despite Equation 5 produces good results and smoother paths in environments with few obstacles, when the environment is cluttered with obstacles, the behavior of the agents are not realistic. To solve this problem, we incorporate the control of the speed in our model, allowing the simulation of agents mood through its magnitude. For instance, a tired agent will probably tend to move slowly whereas an agent that is anxious about its work will tend to move faster. Furthermore, the adjustment of the speed helps to prevent collisions and adds more realism to the simulation [2], e.g., when two pedestrians are in the eminence of collision, they will naturally change its speeds.

[2] Our formalism guarantees that collisions will not happen, however, because the sensor range or/and speed, the agent can perceive another one only when they are about to collide. To avoid abrupt changes in its direction or unnatural movements (see Section 6.3), it can alter its speed according to the collision risk

This consideration is incorporated in Equation 5, producing the equation

$$\mathbf{pos}^t = \mathbf{pos}^{t-1} + \upsilon_{max} f(\mathbf{dgrad}^k, \mathbf{d}^{t-1}) \frac{\mathbf{d}^t}{||\mathbf{d}^t||} \qquad (6)$$

where υ_{max} defines the maximum agent speed and function f generates an output based on the cosine of the angle between vectors $\mathbf{d}^{t-1}$ and $\mathbf{dgrad}^t$, that stops the agent movement or reduces its speed when moving towards an obstacle. Function f is defined as follows.

$$f(\mathbf{x}, \mathbf{y}) = \begin{cases} 0 & \text{if } \cos(\mathbf{x}, \mathbf{y}) < 0 \\ \cos(\mathbf{x}, \mathbf{y}) & \text{otherwise} \end{cases} .$$

If the angle is higher than 180°, then there exists a high hitting probability and this function returns the value 0, doing the agent to stop. Otherwise, the agent speed will change proportionally to the collision risk defined by f. In regions cluttered with obstacles, agents will tend to move slowly. If a given agent is about to cross the path of another one, one of them will stop and wait until the other get through.

6 Results

In this section, the results obtained through the improvements proposed in our path planner are presented. In addition, we present a preliminary result of the extension of our framework for exploratory tasks using multiples humanoids.

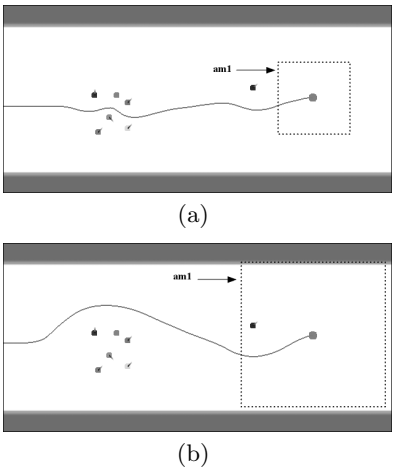


Fig. 6 Varying the size of the agent map

6.1 Analyzing the Agent Displacement

Figure 5 shows some results using Equation 5 without considering variations in the agent speed. The figure shows different paths followed by an agent only varying the parameter η in the interval $[0, 1]$. We assume $\epsilon = 0.7$ and $\mathbf{v} = (0.7, -0.7)$ constants for Equation 1.

As previously commented, the parameter η acts as an inertial factor that tends to keep constant the agent direction insofar $\eta \to 1$. Hence, the bigger the $\eta = 1$ the smoother the path is. In Figure 5e, the influence of η is so strong that the agent has not been able to reach the target position, passing by it and colliding against the wall at the end of the corridor.

6.2 Varying the Size of the Agent Map

Interesting results can be produced in the way agents interact with others in the environment by varying the size of the agent map. The more information on environment is available to the agent the more it will tend to change its behavior to avoid regions with plenty of obstacles. Figure 6 shows two situations where an agent finds a group in its path. In Figure 6a, the side length of the agent map is the half of the corridor width, while in Figure 6b, it corresponds exactly to the corridor width. In the first case, the agent passes in the middle of a group with other agents, whereas in the second case, the agent avoids the group.

The size of the local map can be controlled adaptively using, for example, information about the subjective state of agents. This idea is yet under development, but we obtained preliminary results in robotics context [13] where the robot dynamically adjusts its field of view using information coming from its sonar sensors.

6.3 Varying the Speed

As discussed before, the variation of the speed parameter is very important to generate not only more realistic simulations but also to refine the result of the collision avoidance between agents. Figure 7 shows two experiments that point out the importance of adjusting the agent speed.

In Figure 7a, both agents keep their speed constant during the simulation, tending to follow unnatural paths. For instance, the blue agent described a circular path. In Figure 7b, they vary their speed according to Equation 6, showing a natural balance in the negotiation of the space. The blue agent stops to allow the red one getting through. This reflects more adequately pedestrian behaviors found in real world.

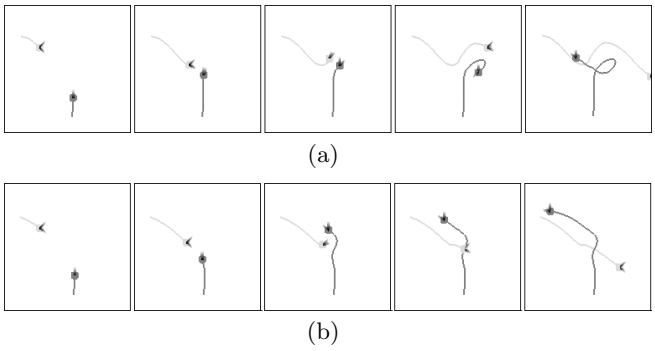


Fig. 7 Simulation of two agents moving one against the other with constant speed (a), and varying the speed (b) to negotiate the space.

6.4 Exploring an Unknown Environment

Our framework is not limited to generate pedestrian behaviors in path planning tasks. It can be also used for more complex tasks as the discovery of targets in unknown environments. In a previous publication [15] we used a small version of this framework to endow a mobile robot Pioneer the ability to seek targets.

Initially, potential fields of the global map cells are updated with a low potential value, indicating that the agent does not know its environment. Then, at each step the agent gathers information using its sensors and adds it into the global map. The cells covered at least one time by the agent sensors have their potential values updated using the relaxation process. The other cells keep their potential value generating an attracting force that pulls the robot towards them. Afterwards, the agent calculates the descent gradient on the cell associated to its current position in the global map and uses it to generate an intermediate goal, comparable to the process commented in Section 4.3. This intermediate goal leads

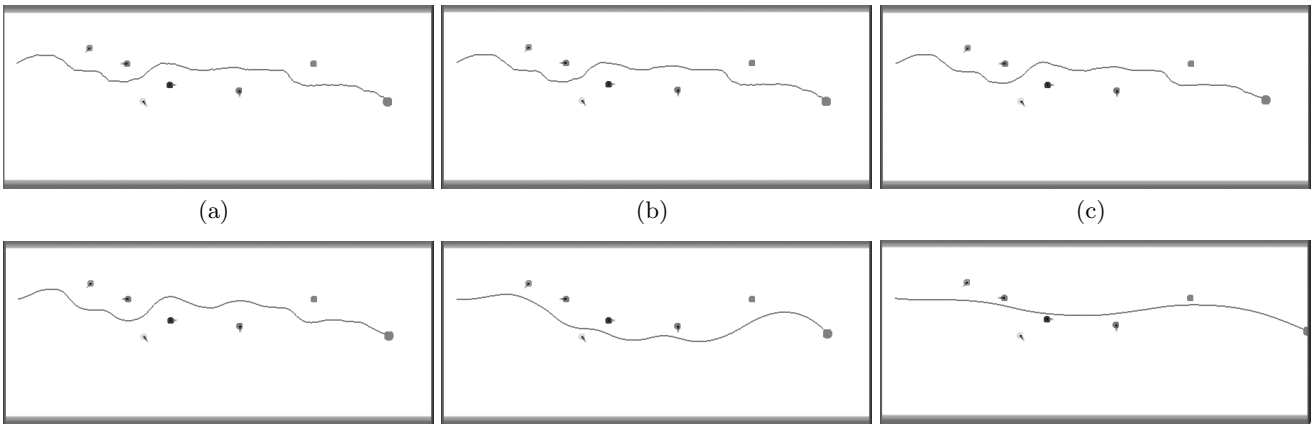


Fig. 5 Varying the parameter η in the interval $[0, 1]$. From (a) to (e), η is equal to 0, 0.25, 0.5, 0.75, 0.95 and 0.99, respectively.

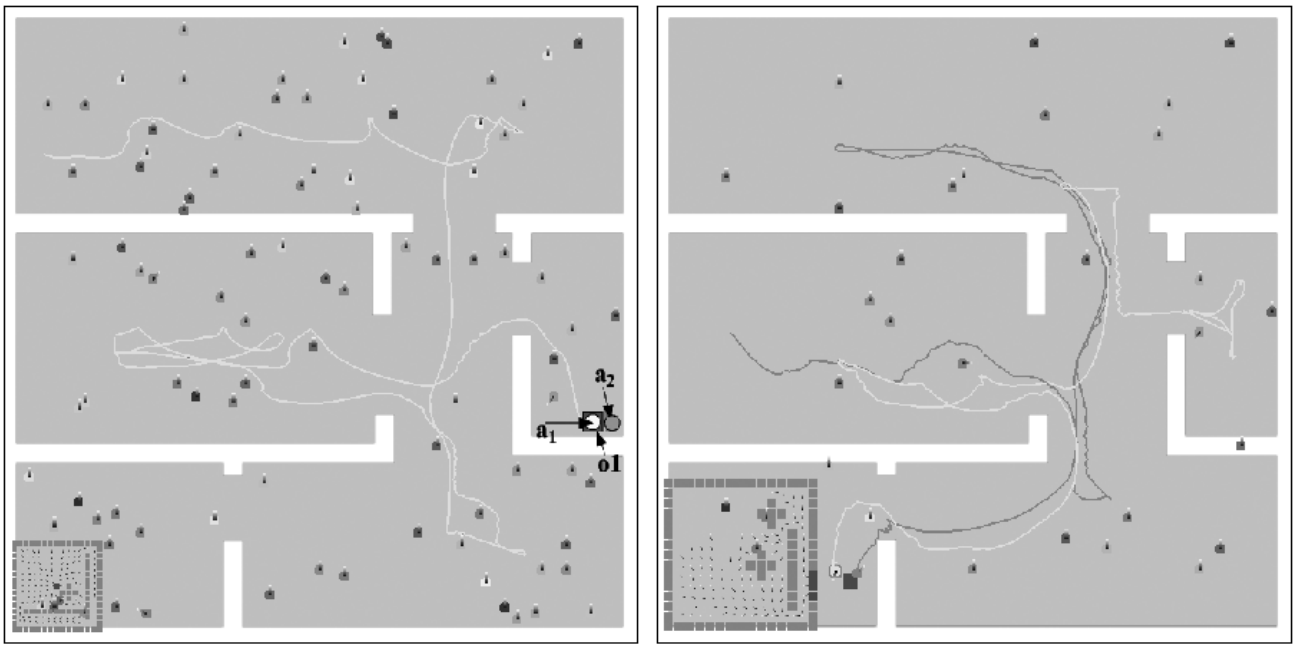


Fig. 8 Searching for an object in an unknown environment: Agent a_1 visits the environment until finding the searched object o_1 (a); two agents a_1 and a_2 are in charge of finding the same object o_1 (b).

the agent automatically towards the nearest region not visited. These steps are very similar to the algorithm proposed by us in a previous work [15].

Figure 8a shows the path followed by the agent a_1 during the search for the object o_1 and Figure 8b shows the case where two agents a_1 and a_2 are searching for the same object o_1. Both situations can be easily found in a party, where a person must find another in a large group of people spatially distributed.

We can perceive that the agent path seems unnatural when compared to the paths generated in the previous experiments. This happens because in the previous examples the global potential field has been computed before the simulation starts and, while in this experiment, it is calculated during the agent movement. Thus, when the agent identifies the presence of an obstacle, it updates the global map and relaxes its cells. After, the system dynamics makes the agent moves to the largest unvisited region, which can be in an opposite direction of the global gradient descent. We are currently treating this situation in order to make the exploratory behavior realistic.

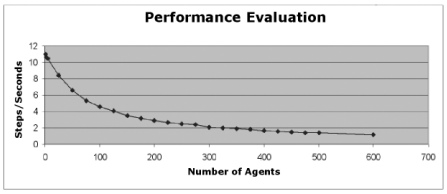


Fig. 9 Performance evaluation.

6.5 Considerations about Performance

Considering the visualization of pedestrian simulations, for each new step the agent do, the motion planner should provide its new position and orientation. According to Mazarakis and Avaritsiotis [9], the frequency of human steps varies from 0.9 to $1Hz$ for someone walking slowly to $3.5Hz$ for someone walking very fast, with a mean of $2Hz$. Then, the performance of a real-time algorithm should be enough to calculate until 3.5 (2 as a mean) steps per second per agent, to animate it and to render the complete scene.

In our experiments, we were not yet concerned by the rendering quality, but only by the quality of the behaviors generated and the number of agents carried by the algorithm. Figure 9 illustrate the results obtained on an ATHLON 64 3500+ 2.21GHz computer with 2.0 GB RAM and graphics card nVidia 7800 GTX. For each step of each agent, considering the mean of 2 steps per second, we have done 60 relaxations of the matrix representing the local map. This allows the management of up to 300 agents concurrently. If we consider the max of 3.5 steps per second, 200 agents are allowed at the same time.

However, this performance evaluation is simplistic, since 3D animation and rendering is not being considered, as well as algorithm optimizations. Besides, a better compromise between rendering, animation and path planning algorithms can be obtained by reducing the number of relaxations for the local maps. In several examples presented in this section, we used 30 relaxations per step done, instead of 60.

7 Conclusions and Future Work

We presented a path planner based on a numerical solution for boundary value problems to produce realistic steering behaviors for virtual humans. In a previous work [4] we demonstrated that adjusting the behavior vector $\mathbf{v}$ and the parameter ϵ, that determines the vector influence, interesting behaviors could be produced.

In this paper, we introduced a new technique to update the position of agents during the simulation, also varying its speed. We proposed a new equation (see Equation 5) to update the agent position that includes the parameter η, representing the inertial factor, used to keep constant the agent direction during the movement. The possibility of varying the speed helps the agents to naturally negotiate the space to try to avoid eminent collisions, as shown in Figure 7. We have also demonstrated that changing the size of local maps (that can be dynamically changed) it is possible to produce different steering behaviors, as illustrated in Figure 6.

Finally, we performed some experiments involving the use of our method to explore unknown environments (see Section 6.4), which can be helpful for applications with very large environments where the topology is not completely known.

The method proposed is formally complete and generates smooth and safe paths. However, as it comes from harmonic functions path planner, it inherits their problems. We minimized the computational cost associated to the convergence of the potential field using, instead of large maps that cover all the environment, small local maps for each agent. Basically, we use several environment maps, one for each target, agent local maps and intermediate goals, as mentioned in Section 4. The size of local maps is a small fraction of the global map size and, therefore, the method has a small amount of cells to calculate the potential field. In this way, it is possible to have several agents acting in the environment still keeping an acceptable running time.

With the conclusion of this first part of the work, we are now exploring the adjustment of our algorithm to manage small groups of agents – while guaranteeing the individual personalities and mood – to be used in applications such as *RTS* (Real-Time Strategy) games. Some efforts will also be made to increase the algorithm performance, such as its adaptation to run into the GPU and the use of efficient data structures, e.g. quadtrees.

References

1. Burgess, R.G., Darken, C.J.: Realistic human path planning using fluid simulation. In: Proceedings of Behavior Representation in Modeling and Simulation (BRIMS) (2004)
2. Choi, M.G., Lee, J., Shin, S.Y.: Planning biped locomotion using motion capture data and probabilistic roadmaps. ACM Trans. Graph. **22**(2), 182–203 (2003). DOI http://doi.acm.org/10.1145/636886.636889
3. Connolly, C., Grupen, R.: On the applications of harmonic functions to robotics. International Journal of Robotic Systems **10**, 931–946 (1993)
4. Dapper, F., Prestes, E., Idiart, M.A.P., Nedel, L.P.: Simulating pedestrian behavior with potential fields. In: Advances in Computer Graphics, *Lecture Notes in Computer Science*, vol. 4035, pp. 324–335. Springer Verlag (2006)
5. James J. Kuffner, J.: Goal-directed navigation for animated characters using real-time path planning and control. In: International Workshop on Modelling and Motion Capture Techniques for Virtual Environments, pp. 171–186. Springer-Verlag, London, UK (1998)

6. Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration space. IEEE Transactions on Robotics and Automation **12**(4), 566–580 (1996)
7. Khatib, O.: Commande dynamique dans l'espace opérational des robots manipulaters en présence d'obstacles. Ph.D. thesis, École Nationale Supérieure de l'Aéronatique et de l'Espace, France (1980)
8. LaValle, S.: Rapidly-exploring random trees: A new tool for path planning. Tech. Rep. 98-11, Computer Science Dept., Iowa State University (1998)
9. Mazarakis G.P., A.J.: A prototype sensor node for footstep detection. In: Proceeedings of the Second European Workshop on Wireless Sensor Networks, pp. 415–418. IEEE Press (2005)
10. Metoyer, R.A., Hodgins, J.K.: Reactive pedestrian path following from examples. The Visual Computer **20**(10), 635–649 (2004)
11. Pettre, J., Simeon, T., Laumond, J.: Planning human walk in virtual environments. In: IEEE/RSJ International Conference on Intelligent Robots and System, vol. 3, pp. 3048 – 3053 (2002)
12. Prestes, E., Engel, P.M., Trevisan, M., Idiart, M.A.: Exploration method using harmonic functions. Robotics and Autonomous Systems **40**(1), 25–42 (2002)
13. Prestes, E., Trevisan, M., Idiart, M.A.P., Engel, P.M.: Bvp-exploration: further improvements. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2003)
14. Reynolds, C.: Steering behaviors for autonomous characters (1999). URL citeseer.ist.psu.edu/reynolds99steering.html
15. Trevisan, M., Idiart, M.A., Prestes, E., Engel, P.M.: Exploratory navigation based on dynamic boundary value problems. Journal of Intelligent and Robotic Systems **45**, 101–114 (2006)