Intelligent Motion Planning for Virtual Characters

Ricardo Sisnett*, Gildardo Sánchez
*Intel GDC, Tecnológico de Monterrey Campus Guadalajara.

Abstract

Artificial Intelligence has developed a number of alternatives to deal with problems that cannot usually be solved with "traditional" algorithms. It is however somewhat surprising that most of the current applications of AI to games are restricted to either A*-based algorithms to find optimal paths or finite automata to represent behaviors. As games are becoming more realistic, other methods should be tried. That is the case of techniques for **automated motion planning** that were originally developed for **robotics** [1]. Such algorithms can be used to generate natural movements for characters in videogames. In this poster we explore the application of a motion planning algorithm that has been proven for robots with many degrees of freedom [4].

Introduction

If we consider a human virtual character as a humanoid robot, then, we can apply algorithms already developed and tested for humanoid robots when dealing with the problem of animating the virtual character. A lot of research in robotics has been concentrated on solving what is called the "basic" problem in motion planning: finding a set of motions that allow a robot to move from a given **start** position to another **goal** position without colliding with the environment [2,4].

Some of the algorithms for motion planning rely on a mathematical representation of the real space, named *Configuration-Space* (or *C-space* for short). That space has as many dimensions as the number of degrees of freedom (dof) of the robot. Computing the *C-space* for a given robot usually takea a lot of time. In practice, the applicability of such methods **is limited to robots with 2-3 degrees of freedom**. If we are interested in the animation of a virtual human character, that is a big problem. A virtual character may have **40-70 dof**. Figure 1 shows a simple character with 46 dof. Despite that, some attempts have been done to apply those methods to the animation of virtual characters, mainly using a hierarchical approach where the computer first solves a **pathfinding** problem for a point in the plane and then the path is transformed in a motion for the virtual character.

A different approach would be to use methods that do not require constructing the C-space. Instead, we could build a discrete representation of the C-space, by means of **sampling** it.

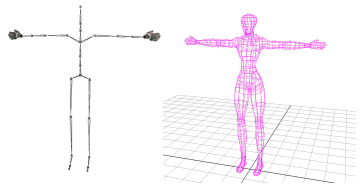

Figure 1. The skeleton of a virtual character in a virtual world.

Every sample represents a **configuration** (pose) of the virtual character, and it can either be classified as free of collision or not. With the samples and the help of what is called a "local planner", a graph or a tree is constructed, and then this data structure is used for searching for a collision-free path for the character.
In this work we will use an algorithm called SBL, which stands for: **Single-Query, Bi-directional Lazy Collision Checking**, described in more detail in [6].

Algorithms

SBL is a planner that searches the free space by building a roadmap made of two trees of configurations (milestones) T_i and T_g. The root of T_i is the initial configuration q_i and the root of T_g is the goal configuration T_g. Every milestone generated during planning is installed in either one of the two trees as the child of an already existing milestone. The link between the two milestones is the straight-line segment joining them in the configuration space. This segment will be tested for collision only when it becomes necessary to perform this test to prove that a candidate path is collision-free.

The algorithm for the planner is:

```
SBL(qi, qg)
1 Install qi and qg as the roots of Ti and Tg,
2 Repeat s times
    3 EXPAND-TREE()
    4 H ← CONNECT-TREES()
    5 If H ≠ nil then return H
6 Return failure
```

Figure 2 shows an example of how the two trees may end connecting after a number of iterations. It is important to mention that although the trees are represented in the plane, the path is computed for all the degrees of freedom. That means that we could **directly** translate the path to a **virtual character**.

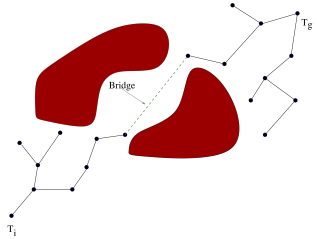

Figure 2. Two trees as built, each starting from initial and goal configurations. The red regions represent forbidden poses due to collisions.

Experiments

In order to test SBL, we implemented the algorithm inside Maya™ using Python. The algorithm requires as input: initial and goal positions of the robot (virtual character) and the geometrical description of the environment, and returns a sequence of positions at each time, which is enough to animate the motion of the virtual character.

Currently, we have two settings, and we will add others soon. The first one was created just to test and debug SBL. Two images are shown in Figure 3. One in the plane (2 dof), and another in the space (3 dof).

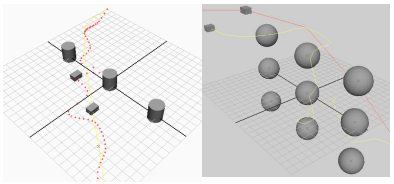

Figure 3. Simple scenario to test SBL.

For the second environment, we took a number of 3D models and created an environment similar to one encountered in a house. Figure 4 illustrates the environment.

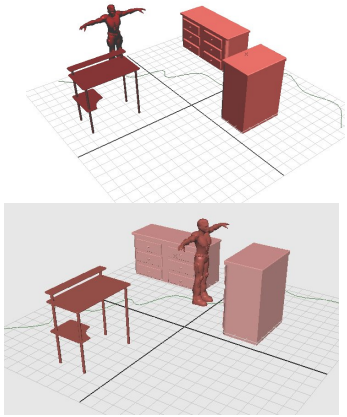

Figure 4. A cluttered scenario with a virtual character. The line on the floor is the path found by SBL.

Conclusions & Future Work

In both environments the planner was able to find a good path in short time (usually a few seconds at the most). That shows that the algorithm can actually be used for games where there could be a good number of obstacles, and the character needs to quickly generate or modify a path.
For the future we would like to add a character with a kinematic structure close to a human, generate a path for the virtual character and use that as the input for a gait animation algorithm and add constraints for the motion like moving obstacles or obstructions for the upper body, to see if SBL is able to generate motions to avoid the collision [3,5]. We will also test it in an environment like the one shown in Figure 5.

Figure 5. A realistic scenario to test SBL.

References

[1]. Arjan Egges, Roland Geraerts, Mark H. Overmars (Eds.): *Motion in Games, Second International Workshop*, MIG 2009, Zeist, The Netherlands, November 21-24, 2009. Proceedings LNCS 5884 Springer 2009.

[2]. H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Boston, 2005.

[3]. M. Lau and J. Kuffner. Behavior planning for character animation. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2005

[4]. J.C. Latombe, Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts, *The International Journal of Robotics Research*.1999; 18: 1119-1128.

[5]. J. Pettré, M. Kallmann, and M.C. Lin. 2008. Motion planning and autonomy for virtual humans. In *ACM SIGGRAPH 2008 Classes* (Los Angeles, California, August 11 - 15, 2008). SIGGRAPH '08. ACM, New York, NY, 1-31.

[6]. G. Sánchez & J.C. Latombe, A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking, R.A. Jarvis and A. Zemlinsky (Eds): *Robotics Research, STAR* 6, pp. 403-407, 2003, Springer-Verlag Berlin Heidelberg.

Acknowledgements

The authors wish to thank the support of Tecnológico de Monterrey-Campus Guadalajara through the Research Chair "Applied Visual Computing" and Intel GDC through Intel Educación-Mexico for all the support provided.Thanks also to Deodato Pechir for providing some 3D models.

Contact information

Gildardo Sánchez Ante
Tecnológico de Monterrey-Campus Guadalajara
gildardo@itesm.mx

Ricardo Sisnett Hernández
Intel GDC, Guadalajara
Ricado.sisnett.hernandez@intel.com