# Combining Path-and-Posture Planning in 3D Environment

G. Zhang
Beijing University of Technology
China

H. Liu
Beijing University of Technology
China

Y. Huang
Institute of Computing Technology
Chinese Academy of Sciences
China

S.M. Zhang
Academy of Mathematics and Systems Science
Chinese Academy of Sciences
China

R.Q. Lu
Academy of Mathematics and Systems Science
Chinese Academy of Sciences
China

***Abstract*--This work is part of a project "full life cycle automation of computer animation generation". Among various animation planning tasks, it is crucial to plan the moving traces of avatars. Our previous animation generation software <Swan> includes a path planning module. However, that module generates only a 2D plan path. In this paper, we introduce a recent work of 3D path planning. Considering the possibility that inadequate 3D environments may request the avatars to undertake special posture while traversing the environment, we combine path planning with posture planning in automatic 3D traverse. Other contributions of this paper include: development a 2D mathematical model for 3D real environment and an algorithm transforming the latter to the former; a new principle of constraint based avatar sensitive path determination; design a calculus for calculating constraint oriented cost calculation based on spatial movement complexity of motion graph clips.**

***Keywords-path-and-posture planning; obstacle map; automatic animation generation***

## I. Introduction

Computer aided animation generation (CAAG) has been a long standing research interest of AI and computer graphics community [1]. Most of these works tried to automate some aspects of animation generation. It was the last two authors of this paper that have been leading a research team from 1990 for exploring an innovative technology--full life cycle automation of computer animation generation [2]. Given a story text written in some limited Chinese natural language (currently Chinese), through the following series of computing processes, the final animation will be generated ultimately:

Step 1: Understanding of the text written in limited Chinese natural language, and converting into a form of semantic representation. Step 2: Analyzing the content of the story with necessary common sense knowledge checks. Step 3: Qualitative scene planning. Step 4: Quantitative scene calculations. Step 5: Cartoon generation supported by an animation knowledge base.

This was the SWAN project supported by the Chinese Academy of Sciences whose first stage was completed in 1995. In its animation generating process, step 3 involves various aspects of planning, such as plot planning, role planning, path planning, etc, where the problem of path planning was not satisfactorily solved. In fact, simply speaking 'path planning' is not enough for characterizing the goal we want to reach. The technique for implementing 'path planning' in automatic 3D animation generation can be divided at least in three levels:

1. 3D motion path (in 3D real environment) planning.

2. 3D path (in 3D Euclidean space) planning.

3. 2D path (in 2D Euclidean space) planning.

That implemented in the early phase of SWAN is only level 3 and is thus only of theoretical meaning. At that time, the basic motion clips were programmed with the motion assembler language script. It was rather cumbersome. Our current goal is to implement level one: 3D motion path-and-posture planning. During the animation generation process, the motion clips will be searched and collected from database dynamically. For implementing this goal, we have the following concerns:

1. To plan a 3D motion path one must find a path on a 3D environment possibly with lots of obstacles such as hills, slopes, caves, rivers, road gaps, etc. All of them have influence on people's walking through.

2. Then, a feasible path for avatar A is not necessary a feasible one for avatar B. A tall guy may climb a hill but a dwarf may not. On the other hand, the latter may climb through a small window but the former may not. As a consequence, the 3D motion path planning is avatar dependent. For a given 3D environment, the computer should derive a path for each particular avatar.

3. Furthermore, even if a path is feasible, the avatar may need to take different postures when passing through different

obstacles. A tall guy may need to bend when walking under a tree while a dwarf does not. Therefore, the 3D path planning must be combined with posture planning.

We report in this paper the following advances made in the new Swan project based on above concerns:

1. Given a 3D environment E and two locations A and B in E, how to determine all possible paths for human beings' walking from A to B?

2. Given the 3D environment E with those possible paths determined above and a particular avatar P, how to determine those places of E which may become obstacles for P's going through, together with a quantitative assessment of obstacles caused by each type of obstacles?

3. Given the above assessment of obstacles, how to determine the feasible paths for P and if possible, the best fit one for P? How should P adjust his posture to overcome the obstacles?

4. Given the best-fit path P and the posture requirements imposed on P, how should the computer get necessary motion clips for P from the motion graph database and combine them to form a complete motion path in 3D animation?

5. If there are not appropriate motion clips for constructing the whole motion path, how should the computer create new compositional motion clips from the existing ones to fill the gap(s)?

## II. Related Work

Much previous work with path planning has only revolved around path. Hofner and Schmidt [3] use the template matching method to compare the current state of the robot with the past states, to find the closest state, and then modify the path in this state to get a new path. In order to improve the adaptability to environmental changing of template matching path planning technology, Shang and Xu [4] design a path planning method that combines environment templates and neural network learning. Template matching path planning method is simple in principle, but the defect of this method is depended on the robot's experiences. Conn and Kam [5] use artificial potential field path planning method to control the robot to avoid obstacles and reach the target position. Obstacles produce repulsive force to the robot. The target point for the robot produces gravity. The resultant force of gravity and repulsive force controls the robot. Artificial potential field path planning technology is simple in principle, but there are usually local minimum points in this method. Tarjan [6] uses visibility graph method to convert a robot to a point. After connecting the point of robot, the target point and the vertices of the obstacles, if no straight lines intersect with any obstacles, a so-called visibility graph is constructed. All paths from the starting point to the target point along these straight lines are collisionless, because any two vertices of a straight line are visible. The aim of path planning is to search the shortest distance from the starting point to the target point through these visible lines. For full coverage path planning, Fu and Lang [7] propose a fuzzy path planning method of moving robot. Zun and Kato [8] propose path planning and collision avoidance methods of moving robots and UAVs based on information fusion technology. Many scholars have done a lot of works on path planning, but they are all limited in the research of posture planning. We searched many relevant literature of posture planning, but only in medical literatures, some described the impacts of different psychological states on the choosing of role's posture. Our job is to combine the path and posture planning to find a path relevant both to the story and the avatars rather than just the shortest path.

## III. Representing Three-Dimensional Scene as a Weighted Two-Dimensional Graph

Path planning is an eternal topic of AI research. But path-and-posture planning has been relatively less studied. The posture of avatar's walking in a 3D environment depends heavily on the condition of the path. Are they walking on a broad way or on a narrow passage? Are they stepping downstairs or upstairs? The postures are all different. To the best of our knowledge, we haven't yet seen such research work on combining path planning with posture planning. So in view of the above mentioned five questions, our work is as follows.

Given a story with a 3D background description D, one of the main steps of automatic animation generation is to transform D into a mathematical graph G before any path planning algorithm can be applied to it. In order to do that, some simplification of 3D background description is necessary.

We consider any non-human object (building, tree, hill, etc) in the environment as an obstacle which limits the motion and posture of avatar. For the moment, we assume that each obstacle takes the form of a polygonal platform. This is not an oversimplification since any 2D region can be approximated with polygons. Moreover, any 3D region can be divided in a finite number of horizontal segments such that each segment can be roughly approximated as a 2D polygonal platform. Thus a 3D region is approximated by layered polygonal platforms with different heights and different polygon forms. A further simplification specifies that only smaller polygonal platforms may be situated on larger ones. Good examples of layered platforms are the terraced fields, figure 1.

FIGURE 1: TERRACED FIELDS.

The process of converting a three-dimensional scene to a 2D graph followed the steps below:

Step 0: Given a 3D environment, figure 3(a).

Step 1: Transform the 3D real environment to a 3D environment model, where each obstacle is modeled with a terraced polygonal form, which is a layered 3D object, where each layer is a segment of some polyhedral cylinder [9]. Horizontal or vertical ladders may exist to connect two neighboring layers of the terraced polygon or to connect two neighboring terraced polygons. Besides, barriers may exist to block the space between obstacles to prevent avatars from walking through the gap, figure 3(b).

Step 2: Transform the 3D environment model to a 2D obstacle map which consists of a set of nested 2D polygons, where 2D nested polygons represent 3D overlapped polygonal layers, while the ladders and barriers are represented by connecting line segments.

Step 3: Transform the 2D obstacle map M to a 2D undirected graph G in mathematical sense as follows, figure 3(c):

First, each polygon P(M) of M produces two nested polygons P(G) of G, where each edge of P(M) produces two parallel edges for P(G) to denote the two parallel paths on and below that polygon edge, figure 2(a);

Next, build new edges for G to connect each vertices of each polygon P(G) with all other vertices of P(G) and also with other polygon's vertices contained in P(G), provided that none of them intrudes in any other polygons, figure 2(b);

Last, build new edges for G to represent any ladders and bridges, where the two end points are situated on both sides of the bridges, figure 2(c); Call the obtained graph C, where all vertices of C are called contact points.

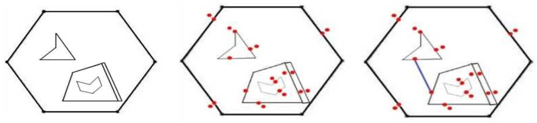
Figure 2(a) Figure 2(b) Figure2(c)

Step 4: Connect the nodes which are at the same height and have a path between them by straight lines but the lines cannot cross any obstacle, and then add the lines into the edge set as edges of the graph. If the path is a broken line, the best one is that each turning point must be at the vertices of an obstacle, instead of turning before encountering the obstacles [10]. In addition, the lines between contact points of different heights are added into the edge set of the graph. As figure 3(d) shows, the blue lines represent the edges of height 0, the red ones are the edges of height 1, and brown dashed lines represent the edges between the contact points on different heights.

Step 5: Remove all unnecessary edges and vertices from C to obtain graph S, figure 3(e). Edges and vertices of C are called unnecessary if they will never be walked through by avatars in any circumstance. For example, any closed polygon CP in C, which does not contain any contact points other than the vertices of its boundary, is unnecessary because no avatar can walk through the boundary (cannot go up or down the boundary), figure 3(f);

Step 6: Assigning constraints, postures and weights to each of the edges. A current table of constraints on avatars is given below in section 4. The postures are determined by a knowledge base with both obstacle type and avatar type as parameters. Part of this knowledge base is shown in table 3. Once the postures are determined, the traverse costs can be calculated. The details of calculating difficulty degree depend on motion graphs and are shown in section 5.

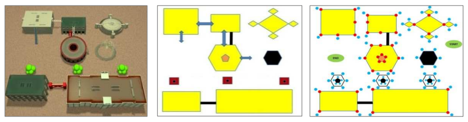
Figure 3(a) Figure 3(b) Figure 3(c)

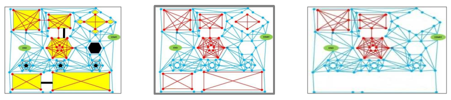
Figure 3(d) Figure 3(e) Figure 3(f)

## IV. Find out All Meaningful Paths

In a three-dimensional environment, the number of paths from the starting point to the end point is countless. If we want to find the best path, the number of candidate paths should be finite. Besides, different from the traditional approaches of path finding which are independent of the individual walking avatars, our approach is individual constraint sensitive. The constraints of the possible paths and the type of the avatar together determine not only the feasibility of walking through any path segment, but also the difficulty (cost) of walking through it. This helps us find all possible paths consistent with constraints from the infinite paths and select the best suitable one from them. Attributes are defined for each obstacle and avatar. Whenever an avatar encounters an obstacle, their attribute values jointly determine the proper posture for the avatar. Based on the key points above, we have designed the attributes of obstacles and avatars as shown by table 1 and table 2 respectively. Table 3 is a matrix that describes an avatar's choice for posture when faced with different obstacles.

TABLE I . The obstacles and its attributes.

| **Obstacles** | **Attributes** | **Appendages** | **Style** |
|---|---|---|---|
| Hillsides | Height, width, slope | The ladders, caves | |
| Caves | Size, length, height | | |
| Bridges | Length, width, height, slope | Handrail | Suspension bridge, single-plank bridge, chain bridge, stairs |
| Trees | Area of the shade, height | | |

TABLE II . The avatars and its attributes.

| **Type** | **Height** | **Width** | **Thickness** |
|---|---|---|---|
| Standard | 160-170cm | 30-40cm | 8-15cm |
| Children | Under 130 cm | Under 20cm | 5-10cm |
| Fatty | 160-170cm | Above 40cm | Above 15cm |
| Tall | 170-190cm | 30-40cm | 8-15cm |

TABLE III . POSTURE CHOICE BASED ON AVATAR AND OBSTACLE.

| | **Hillsides** | **Caves** | **Bridges** | **Trees** |
|---|---|---|---|---|
| **Standard** | Climb | Crawl | Walk | Walk |
| **Children** | Cannot pass | Walk | Crawl | Walk |
| **Fatty** | Round | Round | WF>WB, cannot pass | Walk |
| **Tall** | Climb | Crawl | Walk | Bending over |

Algorithm1 (Find out all meaningful paths):

1. Given an avatar with concrete personal characters (tallness, fatness, age, etc);

2. Extend the graph G with two vertices the starting and ending points of the avatar. They are also contact points and should be connected with other contact points;

3. Check table 3 and remove all edges with constraints which make the avatar's going through impossible;

4. Mark all points as unvisited. Initialize an empty stack. Set the source s as the current point;

5. Mark the CP as visited and push it into the stack;

6. Iteratively check the adjacent points of the CP. For an adjacent point a, if it is the destination, print the path from s to it by printing all the elements of the stack from bottom up; if it is not the destination, mark it as visited, set it as the CP and go to step 2;

7. As for now all the adjacent points of the CP have been visited. Mark the CP as unvisited and pop it out of the stack.

## V. Find the Best Path by Calculating the Posture Costs

According to the requirements of the plot, we search for the best path from them, which is the one that can produce the most well-connected sequence of motion. A salient feature of our algorithms is to use space movement complexity of motion graphs to calculate the costs any avatar has to pay when this avatar is forced to take that posture in some situation.

The popular A* algorithm takes into account only the cost of paths but never the posture of an avatar. There is no obvious interdependence between these two aspects, thus the shortest path does not necessarily be the one with the smoothest posture at the same time. Because we favor visual effects over time saved, our method selects the best path by calculating the degree of well-connectedness for each path [11].

We first convert each path into a sequence of motion clips by selecting the appropriate motion clip for each obstacle on the path and assembling them sequentially. We then measure the well-connectedness of each sequence by averaging the well-connectedness of all consecutive motion clip pairs.

$$\mathrm{D}(f', f) = \min_{\theta,x,z} \sum_{i} \left\| p_i - \underset{\theta,x,z}{T}\, p_i{}' \right\|^2$$

Where the linear transformation $\underset{\theta,x,z}{T}$ rotates a point p about they (vertical) axis by $\theta$ degrees and then translates it by (x, z).

Given the definition of spatial distance, we can define the changing rate $R_C$ of a motion clip $C$ as

$$R_C = \frac{1}{n_C - 1} \sum_{i=1}^{n_C - 1} D(f_i, f_{i+1})$$

Where $f_1$, $f_2$, ..., $f_{n_C}$ are motion clip $C$'s constituent frames.

If a motion clip $C_1$ changes smoothly into the next clip $C_2$, then the spatial distance between $f^1_{n_{C_1}}$, the last frame of $C_1$, and $f^2_1$, the first frame of $C_2$, will be close to the changing rate of $C_1$. However, in most cases, one can expect an abrupt change across two consecutive clips. Thus the difference between this boundary spatial distance and the changing rate of the first clip provides a good measurement of the well-connectedness of the two clips. Formally, for two consecutive clips $C_1$ and $C_2$, their well-connectedness $W_{C_1,C_2}$ is

$$W_{C_1,C_2} = \min\left(0, D\left(f^1_{n_{C_1}}, f^2_1\right) - R_{C_1}\right)$$

Negative values of $D\left(f^1_{n_{C_1}}, f^2_1\right) - R_{C_1}$ indicates that these two clips are well-connected. So when they occur, the well-connectedness $W_{C_1,C_2}$ is set to zero.

We are now ready to define the well-connectedness of a path. For a path $P$ its well-connectedness $\Omega_P$ is

$$\Omega_P = \frac{1}{k-1} \sum_{i=1}^{k-1} W_{C_i,C_{i+1}}$$

whereC1, C1, Ck, are its corresponding sequence of clips.

The path with the lowest well-connectedness is the best path we are searching for. In order to make the best path have more smooth connection, we use linear interpolation algorithm [12].

## VI. Experiments

We made use of the CMU motion capture database, which contains 2,605 trials with each trial being a compound of a dozen unit motions. It is voluminous and should be tailored smartly to meet our purpose.

This is a two-step process, i.e. extraction and standardization. The extraction step extracts proper unit motions from the CMU database. Guided by the description of the trials, a set of candidate trials for a target unit motion is first assembled and then manually scrutinized for its existence. If one is found, it is extracted according to its starting and ending frame numbers. In our application, depending on whether it can be followed by itself, a unit motion is either repetitive or connective. A special treatment is employed for the extraction of the first type. The first frame and the last one of a repetitive motion should be similar enough to ensure the smoothness of the repetition. Manual extraction cannot guarantee this level of similarity. Therefore only the intervals of the first and the last frames in their containing trial are specified manually and a program is responsible for the actual pinpointing of their exact location.

The unit motion thus extracted cannot be used directly. They must be further standardized in two aspects, i.e. alignment and normalization, both of which are performed for the orientation of a root joint. Because of the nature of a 3D place, the direction of a unit motion is required to align to the global coordinate system. The raw unit motions usually do not conform to this requirement and form a slight angle with one of the axes. Alignment is carried out by rotating the root joint, thus the whole body, with respect to the global axis y by this angle. This is equal to multiply a rotation matrix to the left of the orientation which is converted to a quaternion beforehand. After the multiplication the orientation is converted back to Euler angles. Normalization is the process of adding or subtracting proper number of cycles to make sure that each angle of the orientation of a root joint falls into the interval of [0, 360]. This is an essential preparation for the interpolation between two unit motions. With the absence of normalization, the interpolation may whirl the whole body by a full circle during the transition between motions.

The figure 3(a) shows the path of avatar from the start to the end. Figure 3(b), figure 3(c) and figure 3(d) show the motion process of avatar, respectively, showing carefully walking through the overfly, crawling ahead and going down the stairs.

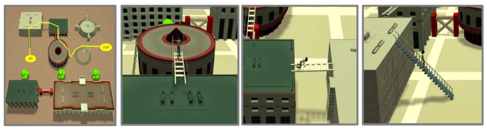

Figure 3(a) Figure 3(b) Figure 3(c) Figure 3(d)

## VII. Conclusions and Future Work

This paper is a part of our “automatic animation generation” project. It proposes a method of combining path-and-posture planning in 3D environment. We represent three-dimensional scene with obstacles map, then we select a set of possible paths from the infinite paths. Next, the paths are converted to motion clips. Finally, according to the similarity metric, we search for the lowest well-connectedness as the best path. Our work, creatively took role’s path and posture planning into consideration according to the requirements of automatic animation generation system, improved the automation degree of generating motion sequences, and enriched the forms of roles’ motions in our system.

However, when planning postures, we could use a more enriched motion database, while study organization motion clips better. More comprehensive consideration of scene and posture information will make the motion selection of a role more consistent with the needs of the system. These will be our future work.

## Acknowledgements

This work was supported by MADIS Key Lab and Sino-Australian Joint-Lab for QCQIP, NSF projects 61232015 and 61073023. The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217.

## References

[1] Lu, R. & Zhang, S., Automatic Generation of Computer Animation. New York/London: Springer, pp.68-69, 2002.

[2] Lu, R., Zhang, S., & Wei, Z., Generate computer animation from natural language stories. Pacific Asian Conf. On Expert Systems, Los Angeles, 1999.

[3] Hofner, C. & Schmidt, G., Path planning and guidance techniques for an autonomous mobile robot. Robotic and Autonomous Systems,14(2),pp. 199-212, 1995.

[4] Shang, Y., Xu, Y. & Pang, Y., AUV global path planning using case based learning algorithm. Robot, 20(6), pp. 427-432, 1998.

[5] Conn, R. & Kam, M., Robot motion planning on N-dimensional star worlds among moving obstacles. IEEE Trans on Robotic Automation, 14(2), pp. 320-325, 1998.

[6] Tarjan, R., A unified approach to path problems. J of the Association for Computing Machinery, 28(3), pp. 577-593, 1981.

[7] Fu, Y. & Lang, S., Fuzzy logic based mobile robot are a filling with vision system for indoor environment. IEEE Int Conf on Computational Intelligence in Robotics and Automation, Monterey.326-331, 1991.

[8] Zun, A., Kato, N. & Nomura, Y., Path planning based on geographical features information for an autonomous mobile robot. Artificial Life and Robotics, 10(2), pp. 149-156, 2006.

[9] Guo, J., Chu, H. & Lu, B., Research on the algorithm for generating minimum-volume encasing box for polyhedron. Computer Applications, 21(1), pp. 38-41, 2001.

[10] Zhang, H., The research and implementation of simulating virtual human walking and path planning in three- dimensional virtual environment. Master's degree thesis, Xiamen University, pp. 51-54, 2009.

[11] Kovar, L., Gleicher, M. & Pighin, F., Motion graphs. In Proceedings of SIGGRAPH, pp. 473–482, 2002.

[12] Watt, A. & Watt M., Advanced Animation and Rendering Techniques theory and practice, Publications: Addison-Wesley Professional, 1992.