Journal of Zhejiang University SCIENCE A
ISSN 1009-3095 (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Automatic target tracking on multi-resolution terrain*

WAN Ming[1], ZHANG Wei[1], MURRAY Marie O.[1], KAUFMAN Arie[2]

([1]*The Boeing Company, P.O. Box 3707, MC 7L-40, Seattle, WA 98124-2207, USA*)

([2]*Department of Computer Science and Center for Visual Computing, State University of New York at Stony Brook, Stony Brook, NY 11794-4400, USA*)

E-mail: {ming.wan; wei.zhang; marie.o.murray}@boeing.com; ari@cs.sunysb.edu

Received Apr. 14, 2006; revision accepted Apr. 29, 2006

Abstract: We propose a high-performance path planning algorithm for automatic target tracking in the applications of real-time simulation and visualization of large-scale terrain datasets, with a large number of moving objects (such as vehicles) tracking multiple moving targets. By using a modified Dijkstra's algorithm, an optimal path between each vehicle-target pair over a weighted grid-presented terrain is computed and updated to eliminate the problem of local minima and losing of tracking. Then, a dynamic path re-planning strategy using multi-resolution representation of a dynamic updating region is proposed to achieve high-performance by trading-off precision for efficiency, while guaranteeing accuracy. Primary experimental results showed that our algorithm successfully achieved 10 to 96 frames per second interactive path-replanning rates during a terrain simulation scenario with 10 to 100 vehicles and multiple moving targets.

Key words: Target tracking, Path planning, Dijkstra's algorithm, Voxel-based modeling, Multi-resolution terrain, Real-time visualization and simulation

doi:10.1631/jzus.2006.A1275 **Document code:** A **CLC number:** TP39

INTRODUCTION

In this paper, we propose an automatic target tracking method in our real-time simulation and animation system of large-scale terrain field. Our system supplies autonomous guidance, navigation and control of moving objects, in order to efficiently perform related challenging missions in manufacture, entertainment, and other real-world applications. Our goal is to deploy a large number of moving vehicles upon a large-scale virtual terrain (essentially a 2D curved terrain surface in 3D space).

A typical mission planning conducted by our simulation system would be organizing a large number of moving vehicles (hundreds or thousands) to efficiently track multiple targets (either static or moving) on a virtual terrain field. Our automatic target tracking method is used to determine the movement of each vehicle based on, for instance, global terrain field information from satellite images. The planned path is capable of rapidly approaching the target or destination, meanwhile cleverly avoiding forbidden zones, and other obstacles. When the targets are moving on large-scale virtual terrain field, our system further provides dynamic tracking of these targets using real-time path planning for each vehicle. These vehicles can be equipped with local sensors and be able to exploit sensor data, so as to precisely locate and recognize targets as well as avoid collision with obstacles. Vehicles may also have self-learning capability during their navigation. All the new information and knowledge from each vehicle will be further shared among other vehicles in the same or different groups, toward a more precise planning and efficient tracking. This paper mainly focuses on the challenging problem of dynamic target tracking and real-time path planning for a large number of moving vehicles on large-scale virtual terrain.

*Project partially supported by NSF (No. CCR0306438) and the Boeing Company, USA

RELATED WORK

The core technique of dynamic tracking of multiple moving targets is essentially motion planning, a crucial research area in robotics. In order to concentrate on the central issues in motion planning, Latombe (1999) distinguished a basic motion planning problem from more general motion planning. He also called the basic motion planning problem path planning, which refers to a purely geometric problem of computing a collision-free path for a robot (a single rigid object) moving in Euclidean workspace among static obstacles. Although such a basic problem is somewhat oversimplified, some mobile robot navigation problems, such as our vehicle movement over terrain, can be perfectly expressed as a realistic instance of it. During the terrain navigation, the positions of the vehicle and the target are considered as the source and the goal of the planned path, while the forbidden zones, constructions, and mountain steeps are considered as static obstacles on terrain. In our simulation system, we also consider other moving vehicles and targets as moving obstacles.

Most of the practical methods for solving the basic motion planning problem are only applicable for single static target within known environment. They can be divided into three categories: roadmap, cell decomposition, and potential field (Latombe, 1991). The roadmap and cell decomposition methods reduce the problem of finding a continuous free path to that of searching a graph (e.g., the visibility graph, the Voronoi diagram, the connectivity graph) by first analyzing the connectivity of the free space. The potential field method was initially proposed for on-line collision avoidance (Khatib, 1986). In this method, the target generates an attractive potential which pulls the robot toward the target, while the obstacles produce a repulsive potential which pushes the robot away from them. The negated gradient of the total potential is treated as an artificial force applied to the robot. Because the gradient depends on the local potential field in the neighborhood of the current robot position, the potential field methods are often called local methods, while the roadmap and cell decomposition methods are called global methods. In comparison to other methods, potential field methods can be very efficient although they have a major drawback: they can get trapped into local minima of the potential function. Also, for dynamic tracking of moving targets, updating the potential field is expensive.

In this paper, we first propose a novel fast path planning method to determine a single moving path from a vehicle toward a target. In this new method, we define the moving path as a minimum-cost path from the vehicle to target in a directed-weighted graph converted from a 2D terrain grid. By assigning different costs to different areas on the terrain, the extracted path can automatically avoid those system-specified or user-specified high cost areas, such as the obstacles and forbidden areas on the terrain.

In particular, we propose to employ the fast Dijkstra's algorithm (Dijkstra, 1959) and a quick heap sorting technique to create a minimum-cost spanning (MCS) tree in the directed-weighted graph in $O(N\log N)$ time, where N is the number of points on the 2D terrain grid. Then, finding the minimum-cost path from a vehicle to a target becomes extracting the corresponding path in the MCS tree, which takes $O(M)$ times, where M is the number of points on the path. Therefore, our efficient path planning can be completed in $O(N\log N)$ time (see details in Section 3.2). Although our method uses costs (or weights) as a potential field method does, these two methods are different. Because our method extracts the path from a global MSC tree, it is a global method and solves the local minima problem. Also, it is more flexible to assign various costs to flexibly avoid various kinds of obstacles on the terrain.

Furthermore, based on this reliable and efficient path planning method, we propose a set of dynamic target tracking methods that are capable of updating the moving paths for hundreds of vehicles at an interactive rate when the targets move. Section 3 describes our automatic target-tracking algorithm, and Section 4 shows experimental results.

AUTOMATIC TARGET TRACKING

Terrain, obstacles, and moving objects representation

The sensory information on the geometry of the terrain and the obstacles are obtained from satellite images, typically including a 2D elevation map and a corresponding color photograph. Based on these sensory images, we generate a high-resolution 2D

terrain grid with the same resolution, forming a virtual digital map of the terrain field (Wan *et al*., 1999). At each grid point, we initially store its elevation and color. Then, we can employ image processing and recognition techniques to identify different objects and areas on the terrain image and mark them on the virtual map. Finally, we assign different costs to grid points located in different areas or occupied by different objects as desired.

For example, an extremely high cost should be assigned to points at forbidden areas so that a planned path can avoid them. Even within a safe area, we may assign costs to points as a function of the elevation gradients at these points to indicate that whether a flatter or a shorter path is preferred. We may also assign a very low cost to each grid point on a road so that we can arrange the vehicles move along a preferred road. Such a global terrain map is shared among vehicles during simulation.

Then, each vehicle sends its own position information or its local sensory target information to other vehicles by making marks on the global terrain map. This information is used both for collision avoidance between moving objects, and updating the terrain map with more detailed or new information at the current terrain field.

Single path planning for a vehicle-target pair

In this section, we describe our novel single path-planning method between a vehicle-target pair. It takes three steps: first, build a 2D directed-weighted graph on the terrain surface; second, create a minimum-cost spanning (MCS) tree on this graph; finally, extract a minimum-cost path from the vehicle toward the target within this MCS tree. We name this new method minimum-cost path planning.

Step 1. Build 2D directed-weighted graph

Each point in the terrain map is defined as a node in the graph, while edges represent the 4- or 8-neighbor relations between the nodes. Each edge has two directions pointing towards its two end nodes, respectively, and each direction has its own cost that has been assigned to the end node it points to.

Fig.1 gives an example of such a directed graph, where A and B are two neighboring vertices on a 2D terrain grid, linked by an edge AB. A and B's weights are $weight(A)$ and $weight(B)$, respectively. W_{AB} and W_{BA} are the weights of directed adges AB and BA, respectively.

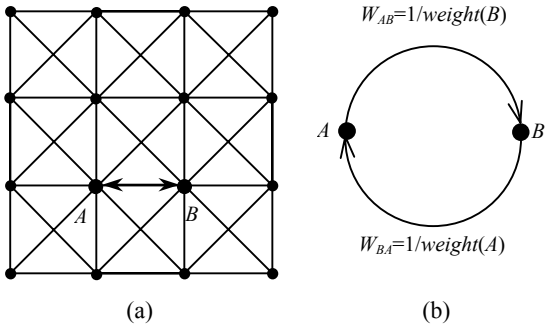

Fig.1 Directed weighted graph converted from 2D terrain grid. (a) Directed weighted graph; (b) Edge AB and its directed weights

Step 2. Create minimum-cost spanning (MCS) tree

We select a source node S at the current vehicle's position, and then rapidly create the MCS tree spanning the entire graph by employing both a modified efficient Dijkstra's algorithm and a fast heap sorting technique (Wan *et al*., 2001).

Specifically, our method consists of the following four steps, where the source point S is at the current vehicle's position:

Step I: Mark source point S, define S as the current node, set its path-link to NULL, and set its $DFS(S)$ to zero, where $DFS(V)$ is the accumulated distance from voxel V to the source point.

Step II: Put each unmarked neighbor, say B, of the current node C into a sorted heap, set its path-link to C, and calculate its $DFS(B)$ as $DFS(C)+DIS(B,C)$, where $DIS(B,C)$ is the distance between B and C.

Step III: Remove the head of the heap, mark it and set it as the current node.

Step IV: Repeat Step 2 and Step 3 until the heap is empty.

A fast heap sorting technique is adopted to find the node with the minimum weight in the current heap by $O(\log N)$ time scale. As a result, our algorithm is completed in $O(N\log N)$ time scale, as compared to the traditional approaches of $O(N^2)$ time scale. Each inside voxel obtains a path-link pointing toward its neighboring voxel, through which it reaches the source point with a minimum weight. It also obtains a DFS-distance to record the length of this minimum DFB-cost path toward the source point. Our minimum-cost spanning tree is represented by the collection of all the path-links. The DFS-distances will help us find the distance between the vehicle and the target.

If a shortest path is preferred, then the DFS-distance can be considered in calculating the accumulated minimum weight in heap sorting.

Step 3. Extract a minimum-cost path

We extract the path in the MCS tree from the source S (the root) to the node at the vehicle's position, according to MCS connectivity. This is the resultant tracking route from the vehicle to the target.

Dynamic tracking of multiple moving targets

The above path-planning method works fine for a one-to-one vehicle-target pair and for static targets only. However, in our system we need to simulate a large number of vehicles with multiple moving targets in the virtual battlefield, which is a much more challenging problem compared to the traditional single vehicle-target pair path planning problem. A brute force approach would be using a single-path planning method to calculate a moving path for each vehicle individually. But the total computational cost would increase with the number of vehicles, which is not affordable. Therefore, we adopt the following three effective strategies to reach an interactive path-updating rate.

1. Multiple-path planning

The first strategy is named multiple-path planning, which takes the vehicles instead of the enemy targets as the goals in path planning. A traditional path planning used to deal with one single source and one single target (at each time frame if the goal is moving). For example, a moving path computed for each vehicle by using the potential field method would be started from the vehicle and ended at the desired target as the goal. However, such a single source (at each vehicle) path planning would be expensive for a large number of sources (vehicles). To solve this problem, we propose a so-called multiple-path planning based on our minimum-cost path planning.

Specifically, we select each target as the source and the vehicles as the goals in our path planning. Then, our minimum-cost path planning can find all the minimum-cost paths toward all the surrounding vehicles at one time. As a result, we speedup the path planning dramatically compared to repeatedly using a single-path planning method on all vehicles. This multiple-path planning procedure is provable to be as fast as a single-path planning procedure from the target to the farthest vehicle. Obviously, as long as the target does not move or moves within a very tiny region, we can keep using these computed paths.

Note that an extra benefit of this method is to automatically identify the total cost from each vehicle to a specific target, which can be accumulated along the corresponding minimum-cost path. Thereby, we can find the lowest cost from a vehicle toward all the targets and assign to this vehicle a target that is reached by this vehicle at the lowest cost. As a result, we can use this strategy to efficiently group the vehicles during the simulation of tracking multiple targets.

2. Variable-range path planning

The second strategy is named variable-range path planning, aiming at tracking a moving target. When a target moves fast, the paths computed by using our multiple-path planning would be updated frequently to keep track of the target movement. However, computation cost for path re-planning at each time frame when a target moves would be expensive and unnecessary. Based on the observation that a vehicle closer to a target changes its path more seriously than the vehicles further away, we propose a variable-range path planning method, which updates the moving path of a vehicle according to both the distances from the vehicle to the target and the amount of movement of the moving target. By using this method, we recomputed the moving paths only for those vehicles within a certain range around the target.

In particular, our path updating is localized within a size-variable area centered at the position of each target. If the change of the position of a target is relatively small compared to the distance between the vehicle and the target, then the change to the path between them is not significant, especially in the segment of the path that is close to the vehicle (Fig.2). Hence, there is no change to the next movement of the vehicle.

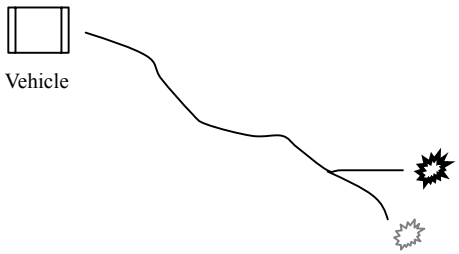

Fig.2 Small movement of target leads to little change to the far away path

Based on the observation above, we propose an adaptive path re-planning strategy for vehicles within a varied range, according to the position of the target. Assume that the updating range is a (R_x×R_z) block centered at (x, z), the coordinates of the position of the target, $k_{\min}$ is a constant, k is an integer, and (Δx, Δz) is the current movement of the target. R_x and R_z is determined as follows:

If

$$\left\{ \exists k \left| \begin{array}{l} (k \geq k_{\min}) \text{ and } (((\Delta x \bmod 2^k = 0) \\ \text{or } (\Delta z \bmod 2^k = 0)) \text{ and } ((\Delta x \bmod 2^{k+1} \neq 0) \\ \text{and } (\Delta z \bmod 2^{k+1} \neq 0))) \end{array} \right. \right\}$$

Then

R_x=min(*TERRAIN_WIDTH*, 2^{k+2}+1),

R_z=min(*TERRAIN_HEIGHT*, 2^{k+2}+1)

Else

R_x=9, R_z=9

Fig.3 gives an example on how to decide the dynamic range. Assuming that $k_{\min}$=2, the paths for vehicles within a (9×9) block centered at the target are updated at every move (every frame). Once a target moves from one (4×4) block to another on the grid, we have k=$k_{\min}$=2. So, we update the paths for those vehicles within a (17×17) block centered at the new position of the target; Furthermore, if the target moves from one (8×8) block to another, then we have k=$k_{\min}$+1=3. So, the updating range is further enlarged to a (33×33) block; and so on. By setting the updating range to be (2^{k+2}+1)×(2^{k+2}+1), this method guarantees that when a target moves out of a (2^k×2^k) block, the vehicles within the (2^{k+2}+1)×(2^{k+2}+1) block are updated.

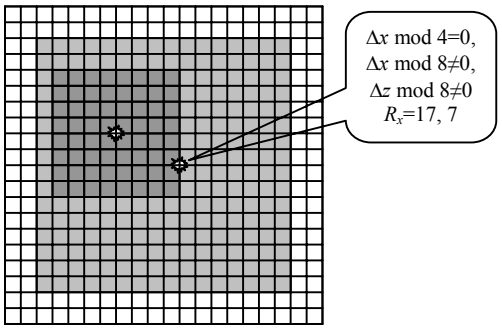

Fig.3 Updating area is always centered at the target. However, its size is decided by the movement of the target

However, note that updating paths only within a fixed range around the target may result in losing of target for those vehicles that never get close enough to the target, hence never get chance to be updated (Fig.4).

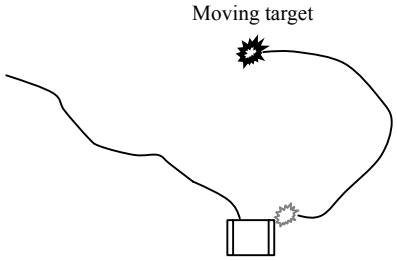

Fig.4 Loses of tracking

This method leaves almost no loss of tracking except for one situation, when there are some obstacles such as mountains or rivers lying between the target and vehicles. In such situations, vehicles could have chosen a longer but flatter route rather than climbing the mountain, according to the assignment of the weights of the edges. To limit the speed of the vehicle or to set $k_{\min}$ to a larger value will reduce the possibility of losing tracking. In order to more effectively and thoroughly solve this problem, while avoiding frequent updating of routes globally, we adopt a passive strategy as follows. We leave the vehicles moving along the route. Once a vehicle reaches the end of the route without finding any target, it sends a rerouting request to its assigned target; then, its target's updating range is forced to be extended to cover that vehicle's location.

3. Multi-resolution based path planning

The third strategy is named multi-resolution based path planning, to further accelerate path planning. Note that the closer a vehicle moves towards a target, the more accurate the moving path should be, so as to make an accurate target tracking. That is also to say, when a vehicle is quite far away from the target, we can trade accuracy for speed. Accordingly, we propose a multi-resolution based path planning which computes the paths for the vehicles on a continuous level-of-detail representation of the terrain. The closer toward the target, the higher resolution terrain data we use. We can employ existing view-dependent continuous level-of-detail object representation approaches, where we use the target to replace the

viewpoint. We can employ either triangle-based (Xia *et al*., 1997; Hoppe, 1998) or voxel-based (Cohen-Or *et al*., 1996; Wan *et al*., 1999) multi-resolution terrain representations. One advantage of the triangle-based representation is that terrain rendering can be accelerated by hardware accelerators.

As an example, we implemented a simple strategy based on our grid representation of large-scale terrain in this paper. If within a $(2^{k+2}+1)\times(2^{k+2}+1)$ block centered at the target, our multiple-path-planning algorithm is applied at a stride of s, then, outside of a block of $(2^{k+2}+1)\times(2^{k+2}+1)$ but within a block of $(2^{k+3}+1)\times(2^{k+3}+1)$, the stride is $2s$. For example, if we assume that $k=k_{\min}$, $s=1$, then the strides within different ranges are illustrated in Fig.5.

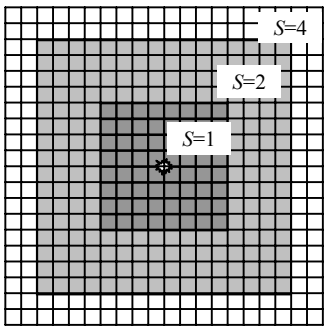

Fig.5 Assume that $k_{\min}=2$. Within the (9×9) block, $s=1$; out of the (9×9) block, but within the (17×17) block, $s=2$, and so on

EXPERIMENTAL RESULTS

To demonstrate the performance of our new automatic target tracking algorithm, we have implemented this algorithm in a large-scale terrain visualization and simulation system. This system was implemented on a 16 R10000 processors SGI Power Challenge.

Fig.6 shows a snapshot of our simulation environment, where 100 vehicles (in green) is tracking 10 moving targets (in red) on top of a 3D terrain volume at a resolution of 512×512×64. There are two challenging requirements to implement such a real-time visualization of the dynamic terrain scene with a large number of vehicles chasing randomly moving targets distributed on different area of the terrain field. One requirement is the real-time rendering of terrain surfaces, which is accelerated by graph-hardware accelerator; the other requirement is the interactive update rates for path-replanning for each vehicle, which is the focus of our work in this paper. Our primary experimented results showed that we have successful reached such challenging goal of an interactive path-replanning rate of around 10 frames per second and above on SGI Challenger, as shown in Table 1 below.

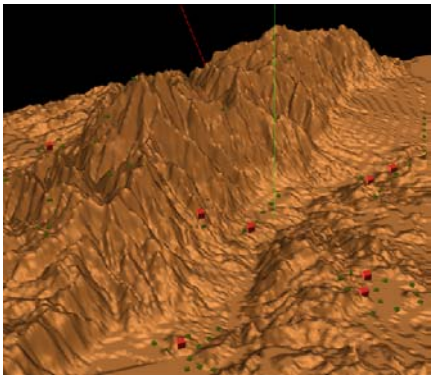

Fig.6 100 vehicles (in green) tracking 10 moving targets (in red) on top of 3D terrain data

Table 1 Automatic path computation time for a large number of vehicles and multiple moving targets

Targets	Time(s)		
	10 vehicles	50 vehicles	100 vehicles
1	0.0102	0.0103	0.0104
5	0.0470	0.0480	0.0490
10	0.0802	0.0848	0.0882

In Table 1, we showed the performance of our algorithm on SGI Challenge with different number of vehicles and moving targets. Our algorithm achieved a satisfactory result of more than 10 frames per second with 100 vehicles tracking 10 moving target. Since the updating time is so tiny at each frame, we measured it through more than 300 consecutive animation frames. An interesting conclusion from our experimentation is that the computation time was dominated by the number of moving targets instead of the number of vehicles. For example, when the number of targets reduced from 10 to 5, the path computation time of 100 vehicles decreased dramatically, and reached an updating rate of 20 frames per second; while, when the number of targets reduced to 1, the path updating rate reached 96 frames

per second. As a result, our algorithm would have a good performance in the scenario of a large number of vehicles (hundreds or thousands vehicles) tracking a small number of moving targets.

Table 2 compares path computation time for 100 vehicles tracking 10 moving target, by using different acceleration strategies described in Section 3.3. By using the brute-force multiple-path planning method in Section 3.3.1, we reached a path updating rate of about 1.1 frames per second. Then, by adopting variable-range path planning method given in Section 3.3.2, we reached about 3.1 frames per second. Finally, by further combining our multi-resolution based path planning method with the two methods above, we successfully reached an interactive updating rate of more than 10 frames per second.

Table 2 Comparison of path computation time with different acceleration strategies, in a terrain scene simulation of 100 vehicles tracking 10 moving targets

Acceleration strategies	Time (s)
Brute force (Section 3.3.1)	0.8926
Variable range (Section 3.3.2)	0.3194
Multi-resolution (Section 3.3.3)	0.0882

CONCLUSION

This paper proposes an efficient automatic target tracking algorithm for a software simulated large-scale terrain field animation and visualization, with specific focus on real-time motion planning for a large number of moving vehicles tracking multiple moving targets. Different from widely used potential field methods, we propose a new minimum-cost path planning that is much more reliable by solving the local-minima problem. Based on this minimum-cost path planning, we further propose a set of effective fast path planning methods, including multiple-path planning, variable-range path planning, and multi-resolution path planning, to achieve interactive updating rates for dynamic path planning of a large number of vehicles chasing multiple moving targets.

In the future, we will extend our algorithms to include flying objects in the 3D space above terrain. In addition, techniques from other areas, such as 3D sensor, image processing, expert systems, and terrain visualization, can also be applied in our simulation system, for better dynamic virtual environment description, target and obstacle identification, robotic vehicle self-learning, and realistic terrain visualization. This system can further be implemented in an immersive virtual environment with, for example, the Workbench, stereo and tracking, to support augmented reality environments.

ACKNOWLEDGEMENT

This work has been partially supported by NSF grant CCR-0306438. Thanks to Aamir Sadiq, Milos Sramek, Qingyu Tang, Suya You, Qi Ke, and other team members of the virtual flythrough project at SUNY Stony Brook.

References

Cohen-Or, D., Rich, E., Lerner, U., Shenkar, V., 1996. A real-time photo-realistic visual flythrough. *IEEE Transactions on Visualization and Computer Graphics*, **2**(3):255-265. [doi:10.1109/2945.537308]

Dijkstra, E., 1959. A note on two problems in connexion the graphs. *Numerische Mathematik*, **1**(1):269-271. [doi:10.1007/BF01386390]

Hoppe, H., 1998. Smooth View-Dependent Level-of-Detail Control and Its Application to Terrain Rendering. Proc. IEEE Visualization Conference, p.35-42.

Khatib, O., 1986. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, **5**(1):90-98.

Latombe, J., 1991. Robot Motion Planning. Kluwer Academic Publishers, Boston, MA.

Latombe, J., 1999. Motion planning: a journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research, Special Issue on Robotics at the Millennium—Part I*, **18**(11):1119-1128.

Wan, M., Qu, H., Kaufman, A., 1999. Virtual Flythrough over Voxel-Based Terrain. Proc. IEEE Virtual Reality Conference, p.53-60.

Wan, M., Dachille, F., Kaufman, A., 2001. Distance-Field Based Skeletons for Virtual Navigation. Proc. IEEE Visualization Conference, p.239-245.

Xia, J., El-Sana, J., Varshney, A., 1997. Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Transactions on Visualization and Computer Graphics*, **3**(2):171-183. [doi:10.1109/2945.597799]