An Auto-Navigation System for Virtual Factories

Tsai-Yen Li
Computer Science Department, National Chengchi University,
Taipei, Taiwan, R.O.C.

Abstract: Interactive 3D graphics is an effective front end for a virtual factory to provide virtual tours, tracking, training, and simulation. With the recent advances in graphics acceleration hardware and in the development of web 3D standards such as VRML, networked virtual reality is becoming a viable and promising man-machine interface. However, it is still a great challenge for a novice user equipped with a regular desktop PC to navigate in most virtual worlds of moderate complexity. We think the main problem is due to the facts that the level of control that a user needs to provide is too low and the display frame rate is not high enough for this servo loop. In this paper, we consider an alternative metaphor of allowing a user to specify locations of interests on a 2D-layout map and let the system automatically generate the animation of guided tours in a virtual factory. Specifically, we aim to generate animations of customizable tour paths and its associated human/camera motions in an on-line manner according to high-level user inputs. We describe an auto-navigation system, in which several efficient path-planning algorithms adapted from robotics are used. The core planning modules in this system are implemented in Java, and common VRML browsers are adopted for 3D visualization of virtual factories.

Keywords: Virtual Guided Tours, Virtual Factory, Path Planning, Camera Motion Planning, Networked Virtual Reality, VRML, and Interactive 3D Graphics.

1. Introduction

Interactive 3D graphics is a crucial man-machine interface for a virtual factory. It provides an effective front end for its users to visualize a virtual factory that may not be accessible in reality. A factory with such an interface will enable its customer to experience a tailored virtual visit at any time. This is a valuable function for factories, such as semiconductor factories, that are not open to the public in general due to their strict clean room policies. In addition, such a system can also be used for internal training, Work-in-Progress (WIP) tracking, process simulation and visualization.

The technologies for interactive 3D graphics or Virtual Reality (VR) are becoming mature. VR is now a popular way of publishing 3D data on the Internet. It is partly due to the ever-growing computing power and 3D graphics acceleration hardware available for personal computers. In addition, the wide acceptance on the standardization of 3D data formats, such as Virtual Reality Modeling Language (VRML)[26], also speeds up the development of networked VR. However, there still exist problems preventing this form of networked VR from being effectively used on a regular desktop PC. For example, manipulating a complex scene with a 2D mouse is still not an easy task for a novice user. A high frame rate is usually required for the user to get familiar with the metaphor. Although a high frame rate can be achieved with a good 3D graphics acceleration card and a carefully designed scene, the computing power can never catch up with user demands on high-quality graphics. As the computing power on PC's increases, a user would expect to be able to navigate in a more realistic (and thus more complex) virtual world. Therefore, we believe that instead of utilizing the full CPU power for graphics rendering only, it might be more rewarding in terms of overall effectiveness to use some CPU cycles for intelligent interface controls.

In this paper we attempt to use machine intelligence as a means to increase the level of control that a user need to provide for effective navigation in a virtual factory. For example, when a user visits a real factory, he/she may only need to express his/her interests by specifying locations on a 2D-layout map. A dedicated tour guide should then plan an optimal route tailored to this visitor and take him/her through these locations of interests. During the customized tour, instead of following the guide's trace, the visitor may follows the tour guide in a more leisurely way such that keeping the visibility with the tour guide is the only requirement.

Although VR can be used for several purposes in a virtual factory, we will focus only on the issue of auto-navigation in this paper. We will describe an implemented networked VR system aiming to provide the tour-guiding service envisioned in the previous paragraph. We describe the main planning modules required to provide the services: *customizable tour planning*, and *intelligent camera positioning*. A user of the system only needs to specify locations of interest on a 2D-layout map and select a traversing mode, the system will generate a good (if not optimal) tour path passing these locations for an animated tour guide to follow in real time. The system will also plan the motion of the camera, representing the user's viewpoint, such that the animated tour guide is always kept inside the view model of the virtual camera.

In the next section, we will present the problems

involved in providing the above services and other related work pertaining to these problems. We give a more formal description of these problems in Section 3 and propose our approaches in Section 4 to solve them in an on-line manner. System architecture and implementation details will then be presented in Section 5. Experimental results and illustrative examples will be shown in Section 6. Conclusions and future extensions of this work are then given in Section 7.

2. Related Work

There have been many VR systems designed to model a real manufacturing environment[22]. Some of them are from the researches in intelligent agents that try to provide various forms of guidance to a user in a virtual environment[6][22]. However, we have not seen many systems that incorporate geometric reasoning, such as path planning or camera tracking, into the application of virtual guided tours. In [7] and [16], path planning are used for generating customizable camera motions while in [11] path planning algorithms are used to generate humanoid animation in an architectural environment. In the scenario described in the previous section, we need to consider two path-planning problems. First, the path followed by the tour guide should be planned in an on-line fashion with a given user specification. Second, the tracking path of the virtual camera representing the user should also be planned according to the tour path. The combination of these two problems and the interactive requirement of the application make it a challenging problem that was not accounted for in the previous systems

The first problem is to generate a good (preferably optimal) tour path that takes the tour guide through all desired locations without colliding with obstacles in the environment[16]. The problem of planning a collision-free path between two configurations is the so-called "Piano Mover's Problem" in the robotics literature, and it has been widely studied in the past two decades[12]. Researches have shown that the general path-planning problem is PSPACE-hard, and its complexity grows exponentially in the number of degrees of freedom (DOF) that the moving object has[21]. When there are several locations to traverse, the problem of determining an optimal traversing sequence becomes the classical "Travelling Salesperson's Problem (TSP)", which is known to be NP-complete[5]. This problem is similar to the vehicle routing problem considered in Operational Research, [3] but it is relatively simpler since it does not impose any capacity or time constraints.

The second problem is about moving the camera in such a way that the tour guide is always kept inside its view cone model. That is, the generated camera motion not only needs to be collision-free with the obstacles in the environment, the view between the camera and the guide should also be clear all the time. There have been several works on generating parametric camera motions[8][25] but only the Intelligent Observer problem defined in robotics [4][9][14] takes this visibility constraint into consideration. In the system described in [14], an optimal tracking path can be computed with a dynamic programming approach. However, for a coarse resolution of workspace, it takes about 20 seconds to generate such a path on a modern workstation. This planning efficiency is not good enough for our interactive application.

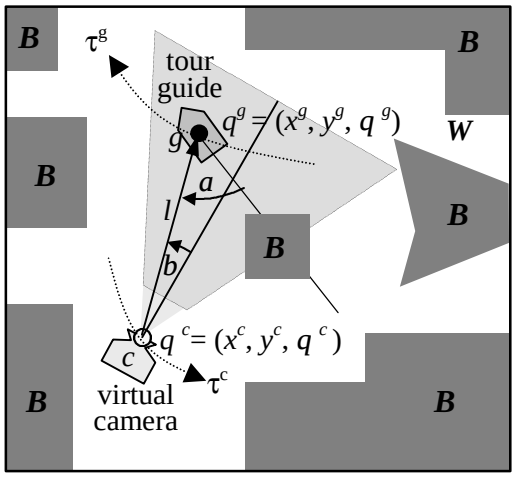


Figure 1: Cartesian-space and view-space configurations of the tour guide and the camera.

3. Problem Formulation

Since computational efficiency is the most important concern in our interactive auto-navigation application, the path-planning problems need to be simplified. For example, since the guided tours considered in our system are all in architectural environments, it is reasonable to assume that objects in the scene can be projected onto a 2D workspace, as shown in Figure 1, where the planning problem is defined.

3.1. The path-planning problems

We assume that both the tour guide and the virtual camera have three degrees of freedom (3-DOF) on a plane, and their geometry can be reasonably represented by 2D polygonal objects. These two objects move in a bounded 2D workspace, *W*, cluttered with polygonal obstacles, *B*. Let the configurations of the tour guide *g* and the camera *c* be denoted by $q^g=(x^g, y^g, q^g)$ and $q^c=(x^c, y^c, q^c)$, respectively. The configuration spaces (C-spaces) for these two objects are composed of all possible configurations of q^g and q^c, respectively[19]. The composite space of these two C-spaces is denoted by *X*, where each composite configuration is denoted by $q=(x^g, y^g, q^g, x^c, y^c, q^c)$. The objective of the auto-navigation system is to generate a legal path *t* in *X* connecting an initial configuration q_i and a final configuration q_f. A configuration is legal if and only if *g* and *c* do not overlap with *B* or with each other, and *g* is inside the view cone of *c* at any time as shown in Figure 1.

The solution to our tour-planning problem resides

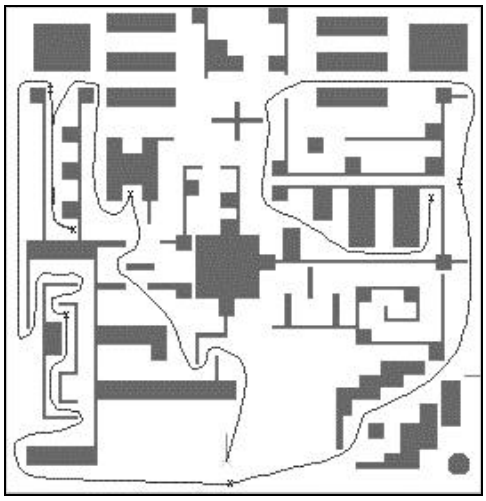


Figure 2: a sample tour plan for a virtual factory that traverses all user-specified locations marked with x. (Obstacles are in grey.)

in this composite C-space defined above. However, it is not likely that we can design an on-line planner that can solve the problem in a space of such high dimensionality. (Most on-line planners exist for problems with less than 4-DOF[17].) Fortunately, in our auto-navigation problem, the motions of the tour guide and the camera are not without dependency. The path for the tour guide should be determined before the camera path. Therefore, a reasonable approach would be to plan a tour path t^g first and then generate the camera path t^c according to t^g.

After t^g is determined, we can plan the camera motion by connecting them with an imaginary line segment dragging the camera as the tour guide moves. Therefore, instead of planning in the camera's Cartesian space (x, y, q), we re-parameterize the camera configuration with an equivalent representation, (l, a, b), relative to the tour guide, as shown in Figure 1. These parameters, defined simply for planning convenience, are called *view distance*, *tracking angle* and *view angle*, respective. Since t^g is a function of time, the planning space for c now becomes a 4D configuration-time space (CT-space) defined by (t, l, a, b).

3.2. The optimal sequence problem

One of the challenges in this auto-navigation system is on generating an optimal traversing sequence through the user-specified locations of interests. An illustrative path traversing through all user-specified locations is shown in Figure 2. This problem belongs to the classical NP-complete TSP problem, where no efficient algorithms are found so far. Nevertheless, there exist approximate algorithms that can find near-optimal solutions in polynomial time if the Euclidean distances between locations can be determined[5]. However, for a workspace cluttered with obstacles, the arc length of a collision-free path is the real distance between two locations, and this information is not known prior to the search process. Therefore, it is difficult to find an optimal solution in general. However, the auto-navigation problem under consideration may still contain some characteristics that allow an efficient implementation for sub-optimal solutions.

4. The Proposed Approaches

In this section, we propose our practical approaches to the computationally difficult problems described in the previous section. An objective of these approaches is to bring the planning activities on-line for interactive applications. Simplifications are required to achieve this goal but we will show that through careful design and appropriate simplifications, the results could still be quite satisfactory in terms of efficiency and quality.

The planning approach we take in this paper is similar to the distributed approach described in [2]. The planning spaces are represented by uniform grids of an appropriate resolution, and the planning algorithms are variations of the best-first search algorithm with different "best" criteria.

4.1. Decoupled approach for path planning

As mentioned in the previous section, although the overall planning space is a six-dimensional space, the dimensions are not of independent. Therefore, we take a decoupled approach by decomposing the problem into two smaller sub-problems of planning for two objects of 3-DOF each and solving them in sequence[17]. That is, we first plan the tour path in the tour guide's C-space (3D) and then plan the camera path in the camera's CT-space (4D) with the previously planned tour path as a constraint.

There exist efficient algorithms in robotics that can generate collision-free paths in fractions of a second for problems of less than 5D [2][17]. However, in the worst-case situations the running time could still be tens of seconds or even minutes. In addition, the length of the overall path in our application is typically much longer than the one considered in robotics. Therefore, further simplifications or decompositions are needed for better on-line performance. For each of the two sub-problems above, we further decompose their search space according to the moving object's geometric characteristics.

For the tour-planning problem, we assume that the geometry of tour guide can be approximated by an enclosing circle. We can then reduce the dimensionality of the C-space from three to two due to the geometric symmetry. The C-space can be explicitly computed by growing the workspace obstacles with the radius of the enclosing circle[13]. This is a conservative but reasonable assumption because in our application a planned tour does not normally guide a visitor to pass through narrow passages. Tour paths are generated in the reduced C-space consisting of 2D configurations (x, y) only. We add the decoupled q

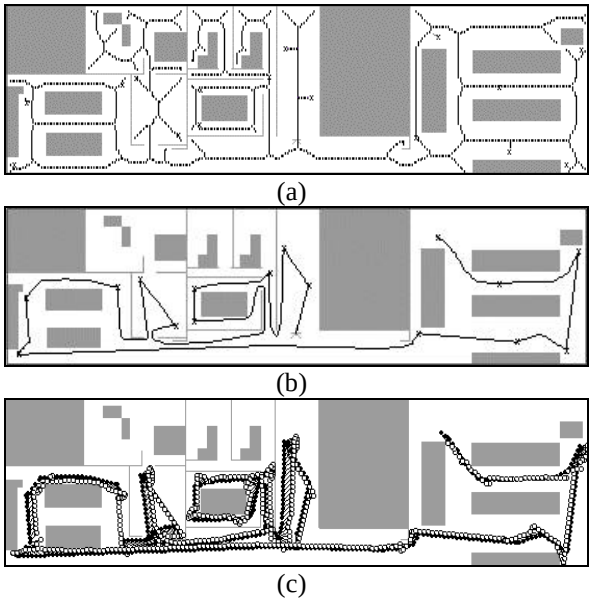


Figure 3: an example of (a) freespace skeleton, (b) tour path, and (c) the corresponding camera path, where the guide and the camera are depicted as solid dots and hollow circles, respectively.

component of each configuration back into the path in a postprocessing step. In this step, we assume that a tour guide always faces the tangential direction of the path. At the locations where large discrepancies in q occur, we replace the path segments of sharp turns with smooth curves and add additional steps into the path to make the rotational transition smooth.

For the camera-planning problem, we also take a decoupled approach to reduce the dimensionality of the search space. For the CT-space (t, l, a b) defined in the previous section, we lock the view angle, b, between the camera and the tour guide such that the tour guide is always at the center of the camera's field of view. With this assumption, we then search for a collision-free path in the reduced CT-space connecting an initial configuration $q_i=(t_i, l_i, a_i)$ to any legal final configuration $q_f=(t_f, *, *)$, where * represents any legal values. During the search, the configuration in this CT space must be advanced in time, and the movement should be confined to some maximal velocity constraint. Since planning efficiency is the main concern of our application, time difference from the final time slice is taken as the primary criterion for the best-first search algorithm. Thus, the planner will return the first feasible path it finds during the search. After a feasible tracking path has been found, we unlock and fill in the b dimension in such a way that the camera rotations are minimized. More detail description of the tracking motion planning can be found in [18].

4.2. Greedy approach for optimal sequencing

The difficulties of determining an optimal traversing sequence in our problem lie on the unknown distances between locations and the large number of possible traversing orders in general. We overcome these difficulties by taking a greedy approach which traverses on a skeleton (discrete Voronoi diagram) built in the free space of the aforementioned 2D C-space. An example of the skeleton is shown in Figure 3(a). The skeleton captures the topological information of the freespace and can be computed automatically for a workspace bitmap in a preprocessing step[12]. All locations to be traversed are connected to the skeleton on the fly and treated as part of the skeleton before the planning starts.

Unlike the classical path-planning problem where the initial and goal configurations are well defined prior to planning, we do not have a definite goal to reach in our problem. We start the search from the initial configuration and allow the search to propagate in a breadth-first fashion along the skeleton until it reaches a nearest goal location that has not been visited before. In other words, the traversing distance along the skeleton is the search criterion for the best-first search algorithm. We repeat this process until all locations are visited. Since the search is limited on the skeleton, the search space is considerably reduced. Although the resulting tour path is not guaranteed to be optimal, this greedy approach works pretty well in the examples we have tested. Examples of the tour path and the camera path generated using these approaches are shown in Figures 3(b) and 3(c), respectively. In Figure 3(c), the tour guide (solid dot) and the virtual camera (hollow circle) are connected by a line segment, which remains collision free with the obstacles all the time along the tour. Detail planning parameters and efficiency will be reported in Subsection 6.2.

5. System Architecture and Implementation

In this section, we describe how we implemented the planning or simulation modules and how these different modules work together to provide the auto-navigation service.

5.1. System architecture

The system is comprised of five logical modules as depicted in Figure 4: VRML display module, control module, tour-planning module, camera-planning module, and humanoid simulation module. The humanoid simulation module is for generating the human walking gaits of various sizes in real time according to the tour path. Only the VRML and tour-planning modules contain graphical user interfaces. A user specifies the locations of interests by pointing directly on a layout map presented by the tour-planning applet. The user can also specify preferences such as using default locations, choosing a preferred traversing order, etc. through this interface. The final guided tour is presented to the user via a common VRML browser in a regular web page.

A typical data flow in the system, as shown in

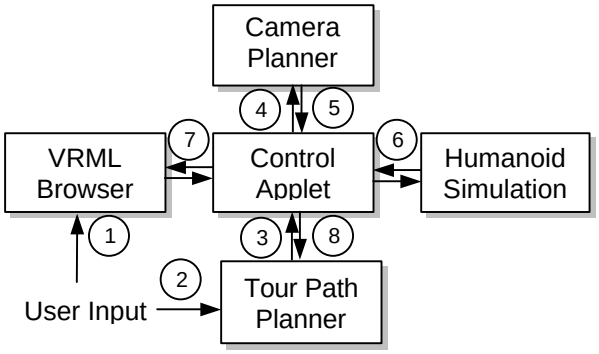
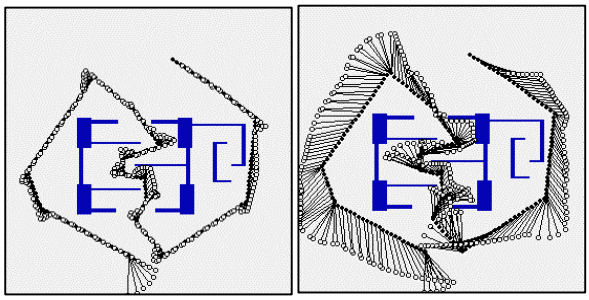


Figure 4: system architecture and data flow among different modules in the tour-guiding system



(a) (b)

Figure 5: Examples of camera paths (circles) for the same tour path (dots) but with different search criteria emphasizing on a and l, respectively

Figure 4, is described as follows.

(1) manually navigating in the VRML browser,
(2) specifying locations of interests via the 2D interface and starting to generate the customized tour when ready,
(3) passing the generated tour path to the control module,
(4) passing the tour path to the camera planner for generating the camera motion,
(5) returning the camera motion back to the control module,
(6) passing the tour path to the humanoid simulation module to generate appropriate joint motions for walking simulation,
(7) passing the motions of the tour guide and the camera to the VRML browser to create animation of the virtual tour,
(8) updating the locations of the tour guide and camera in the 2D interface.

The animation stops when the system has finished traversing all desired locations.

5.2. *Implementation*

Except for the VRML display module, all modules described in the previous subsections are implemented as Java applets. These applets communicate with each other through the scripting model of the hosting web browser. The control applet communicates with the VRML browser via the External Authoring Interface (EAI) [20].

Although the 2D floor map used in the planners can be generated directly from the VRML models, it is currently read in from a data file created separately. The dimensions of the tour guide used in the walking gait simulator are computed according to the standard humanoid model in the VRML file. This model is obtained dynamically via the EAI module and can therefore be easily replaced with other avatars constructed in accordance with the humanoid specification[24].

6. Experimental Results

All of the running times reported in this paper were taken on a 450MHz Pentium II PC running MS Windows NT. In the current system, we have used our departmental building as an example to demonstrate the efficiency and usability of the system. The same planning modules are being tested with more virtual factory models such as the one shown in Figure 5. Note that in order to remain behind the tour guide while making turns in narrow passages, the camera needs to follow the tour guide closely from time to time.

6.1. *Graphical user interface*

The system provides a graphical user interface in a web browser running on regular desktop PC's. A snapshot of the graphical user interface of the system is shown in Figure 6. The VRML browser is at the top while the 2D floor map is shown at the bottom. The snapshot was taken at our personal computer laboratory. The size of the VRML file is about 1.3MB and the frame rate is about 0.5 to 4 frames per second. On a regular PC, it is indeed a pain to navigate in a virtual scene of this complexity. However, this is exactly the main reason that motivates our auto-navigation system in the first place. During the animation of the tour, a user can interact with the animation directly on the VRML browser or through the 2D interface. In particular, in the 2D interface (at the lower right corner of Figure 6) a user can adjust the head orientation horizontally or vertically independent of the body motion (which is always on a plane).

6.2. *Planning efficiencies*

The search spaces of all path planners in this system are represented as multidimensional bitmaps. The workspace bitmap resolution used in the tour planner is 560x150 while the maximal allowable rotation in a walking step is 10 degrees. In the camera planner the resolution for the last two parameters (l and a) in the CT-space (t, l, a) is 50x100, and the resolution in the t dimension depends on the number of steps in the path generated by the tour planner. In the example shown in Figure 3, there are 1112 steps in the overall tour path. The running times for the tour planner and the camera planner are 0.19 and 0.38 seconds, respectively. The tour guide's animation is generated on

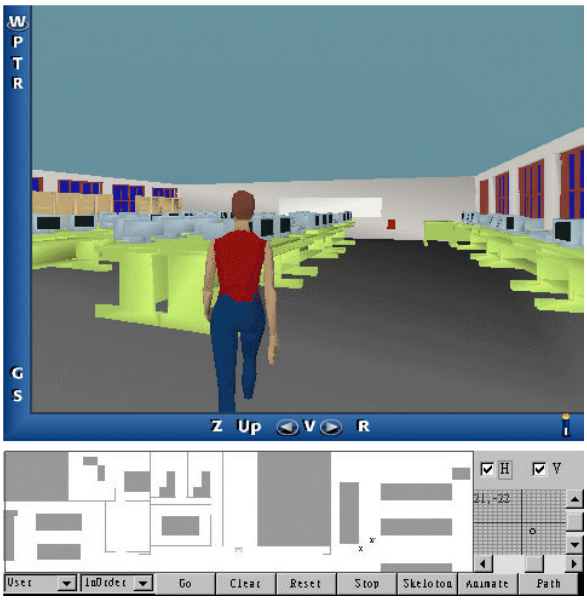


Figure 6: a snapshot of the graphical user interface in the touring system

demand. The average running time for the humanoid simulator to generate one walking step is about 3 ms, which is negligible during the animation.

In summary, the overall planning time of the system is quite satisfactory for our interactive application. After the user selects locations of interests and clicks on the "go" button, complete paths for the tour guide and the virtual camera can be ready for execution in fractions of a second to a few seconds.

7. Conclusion and Future Extensions

Although networked virtual reality is a promising direction for realistic experiences in a virtual world, there are several problems to overcome in order to make 3D navigation a more pleasant experience. We believe that the key will lie on using machine intelligence to bring up the level of control that a user needs to provide. In this paper, we described an implemented system capable of providing customizable guided tours for a remote user to visit a virtual factory. This system incorporates motion-planning algorithms to generate collision-free paths for the tour guide and for the virtual camera. The motions for the tour guide and the camera are then animated in a 3D VRML browser. Experiments have shown that the generated guided tours including the human animations are all very effective, and the computation times for each of the planning or simulation modules are adequate for on-line applications.

In the current system, the only goal of the tour planner is to pass through all locations specified by the user. However, the purpose of taking a guided tour in a virtual world might be more than being there. For example, the tour guide may stop at a popular tourist spot and provide the user with richer contents in forms of body gestures or narration. The user may also choose a different tracking strategy that allows the tour guide to leave its field of view at different occasions for certain amount of time. We are in the process of capturing these Cinematographical idioms in a finite state machine and allowing them to be incorporated into the tour path[10].

The 2D simplifications made in the current system are adequate for the tour guiding purpose, but there are situations where full 3D camera motions might be preferred or even required for tracking an arbitrary objects, such as a Work-in-Process (WIP), in a virtual factory. The computation cost for 3D interference detection is much higher than the 2D case. In addition, the search space for full 3D motions is so large that a complete planning algorithm might not be feasible. To deal with this challenge, we are experimenting with a random sampling scheme that has been shown effective for robot path planning with high-dimensional configuration space[1].

Although the traversing sequence found with our greedy strategy are all quite satisfactory, an optimal sequence is still desirable if it can be found in a short amount of time. We are in the process of designing an algorithm taking advantages of the topological characteristics of the skeleton in an architectural environment. We observe that the structure of the skeleton in a typical floor map is not a dense graph. Instead, it is more like a tree structure except for small local cyclic loops. If we can isolate and process these loops separately, an optimal solution can be found by composing the optimal solutions from these general but smaller TSP problems.

Acknowledgements

This work is supported by grants from National Science Council under NSC 88-2218-E-004-002.

References

[1] J. Barraquand, L. Kavraki, J.C. Latombe, T.Y. Li, and P. Raghavan, "A Random Sampling Scheme for Path Planning," in *International Journal of Robotics Research*, 16(6), P759-774, December, 1997.

[2] J. Barraquand and J.-C. Latombe, "Robot Motion Planning: A Distributed Representation Approach," *The International Journal of Robotics Research*, 10(6), pp. 628-649, December 1991.

[3] J.E. Beasley and N. Christofides, "Vehicle Routing with a Sparse Feasibility Graph," in *European Journal of Operational Research 98*, pp. 499-511, 1997.

[4] C. Becker, H. Gonzalez-Banos, J.-C. Latombe, and C. Tomasi, "An Intelligent Observer," in *Proceedings of International Symposium on Experimental Robotics*, pp. 94-99, 1995.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, "*Introduction to Algorithms*," The MIT press, pp.969-974, 1994.

[6] P. Doyle and B. Heyes-Roth, "An Intelligent

Guide for Virtual Environments," *Proceedings of the First International Conference on Autonomous Agents*, pp. 508-509, February 1997.

[7] S. Drucker and D. Zelter, "Intelligent Camera Control in a Virtual Environment," in *Proceedings of Graphics Interface' 94*, 1994.

[8] S. Drucker, and D. Zelter, "CamDroid: A System for Implementing Intelligent Camera Control," in *Proceedings of Symposium on Interactive 3D Graphics*, 1995.

[9] H. H. Gonzalez-Banos, L. Guibas, J.-C. Latombe, S.M. LaValle, D. Lin, R. Motwani, and C. Tomasi, "Motion Planning with Visibility Constraints: Building Autonomous Observers," in *Proceedings of the Eighth International Symposium of Robotics Research*, Hayama, Japan, October 3-7, 1997.

[10] L.-W. He, M. Cohen and D. Salesin, "The Virtual Cinematographer: a Paradigm for Automatic Real-time Camera Control and Directing," *Computer Graphics Proceedings,* SIGGRAPH95, pp. 217-224, 1995.

[11] J.J. Kuffner. "Goal-Directed Navigation for Animated Characters Using Real-Time Path Planning and Control". In *Proceedings of CAPTECH '98: Workshop on Modeling and Motion Capture Techniques for Virtual Environments*, Geneva, Switzerland, November 1998.

[12] J.-C. Latombe, "*Robot Motion Planning*," Kluwer Publishing, pp. 318-334, 1990.

[13] J.P. Laumond, "Obstacle Growing in a Non-Polygonal World," *Information Processing Letters*, 25(1), pp. 41-50, 1987.

[14] S. M. LaValle, H. H. Gonzalez-Banos, C. Becker, J.-C. Latombe, "Motion Strategies for Maintaining Visibility of a Moving Target," in *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, 1997.

[15] S. Lavallee, J. Troccaz, L. Gaborit, A.L. Benabid P. Cinquin, and D. Hoffman, "Image-Guided Operating Robot: A Clinical Application in Stereotactic Neurosurgery," in R.H. Taylor, S. Lavallee, G.C. Gurdea, and R. Mosges, editors, *Computer Integrated Surgery*, pp. 343-351, MIT Press, Cambridge, MA, 1996.

[16] T.-Y. Li, L.-K. Gan, and C.-F. Su, "Generating Customizable Guided Tours for Networked Virtual Environments," in *Proceedings of 1997 National Computer Symposium (NCS'97)*, Taichung, pp. D189-D195, December 1997.

[17] T.-Y. Li and J.-C. Latombe, "Online Manipulation Planning for Two Robot Arms in a Dynamic Environment," in *International Journal of Robotics Research*, 16(2):144-167, April, 1997.

[18] T.-Y. Li and T.-H. Yu, "Planning Object Tracking Motions," in *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, May 1999.

[19] T. Lozano-Perez, "Spatial Planning: A Configuration Space Approach," *IEEE Transaction on Computers*, 32(3), pp108-120, 1983.

[20] Chris Marrin, "*Proposal for a VRML 2.0 Informative Annex: External Authoring Interface Reference*," URL:http://www.vrml.org/WorkingGroups/vrml-eai/ExternalInterface.html, November 21, 1997.

[21] J.H. Reif, "Complexity of the Mover's Problem and Generalizations," *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pp. 421-427, 1979.

[22] S. Resseler, A. Godil, Q. Wang, G. Seidman, "A VRML Integration Methodology for Manufacturing Applications", in *Proceedings of Virtual Reality Modeling Language (VRML99)*, 1999.

[23] J. Rickel and W.L. Johnson, "Integrating Pedagogical Capabilities in a Virtual Environment Agent," *Proceedings of the First International Conference on Autonomous Agents*, February 1997.

[24] "Specification for a Standard VRML Humanoid," http://ece.uwaterloo.ca/~h-anim, September 10, 1998.

[25] S. Teller, and C. Sequin, "Visibility Preprocessing For Interactive Walkthroughs," in ACM *Computer Graphics (Proceedings of SIGGRAPH'91)*, 25(4):61-69, 1991.

[26] "VRML97 Specification," International Standard ISO/IEC 14772-1:1997, http://www.vrml.org/Specifications/VRML97.

Contact

Tsai-Yen Li
Associate Professor
Computer Science Department
National Chengchi University
No 64, Sec.2, Chih-Nan Rd,
Taipei, Taiwan, R.O.C.
Phone: 886-2-29387642 Fax: 866-2-22341494
Email: li@cs.nccu.edu.tw