Planning Tracking Motions for an Intelligent Virtual Camera

Tsai-Yen Li
Computer Science Department
National Chengchi University,
Taipei, Taiwan, R.O.C.
li@cs.nccu.edu.tw

Tzong-Hann Yu
Computer Science Department
National Chengchi University,
Taipei, Taiwan, R.O.C.
s8410@cs.nccu.edu.tw

Abstract

We consider the problem of automatically generating viewpoint motions for a virtual camera tracking a moving target. Given the target's trajectory, we plan the motion of a camera to maintain visibility with the target while optimizing certain camera-specific criteria, such as preferred view angle or view distance. Previous work has shown that optimal paths for this problem can be found with a dynamic programming approach. However, efficiency for this type of approach, even with a simplified view model, is not good enough for on-line applications. We proposed a novel approach that searches for a collision-free path satisfying visibility constraints in the relative configuration space between the moving target and the tracking camera. The search is conducted in a best-first fashion according to time and other optimization criteria. We have implemented this algorithm in a software package that is capable of generating such a tracking motion in fractions of a second. This software module is part of a virtual presence system that generates customizable guided tours to assist users in navigating a virtual scene.

1. Introduction

Tracking a moving object with a continuously non-obstructive view is a challenge task for humans as well as for computers. For human professionals, it is the type of the problems faced by cinematographers for over a century. For computers, maintaining visibility with an object is also crucial in many applications. For example, in medical applications, it is important to automatically control camera motions to present a sequence of non-occlusive images to surgeons during a surgery [11]. Similar requirements also exist in telepresence applications, where visual feedback is usually the primary means for gathering information [1].

The problem considered in this paper was motivated by a virtual presence application in a virtual factory. In such an application, keeping track of a moving object with a virtual camera is an essential function for its user. For example, a target of interest can be a customized tour guide in the virtual world as proposed in [12]. Such a system aims to help a remote visitor navigate through a 3D virtual environment by letting the visitor specify locations of interests on a 2D-layout map. According to this specification, the system will automatically generate a customized guided tour, including an animated human tour guide, and take the visitor through these locations in a desirable sequence. In this paper, we specifically consider the problem of automatically generating camera motions according to the pre-planned trajectory of the tour guide. We shall describe an implemented motion planner that is capable of planning a collision-free motion for the visitor's viewpoint (alternatively, the camera) such that the tour guide is always kept in sight.

The problem we consider is different from the one in traditional active vision systems in robotics. In such systems, cameras are controlled according to pre-specified rules and visibility of the target is not guaranteed. In our problem, we would like to maintain visibility all the time by planning a feasible camera motion based on the known trajectory of the target. This problem also differs from the traditional path-planning problem, where geometry interference is the only concern. Here, we have to consider additional visibility constraint, and the obstacles that obstruct a camera's geometry and its view are not necessarily the same.

1.1. Related Work

Researches pertaining to visibility computation can be found in the literature of computer graphics and robotics. In computer graphics, visibility information can be used for geometry culling in interactive 3D graphics such as building walkthrough [18] or be used to facilitate illumination computation [17]. In computer animation, a camera can be controlled to satisfy constraints such as fixing points in a image space[3] or be directed in real time according to the film idioms (heuristics) commonly used in Cinematography[6]. In robotics, visibility-related problems include placing sensors at appropriate locations to ensure the visibility of certain objects. Installing a minimal number of sensors in a static manner is the so-called art gallery problem[15] while allowing the sensors (usually cameras) to move actively is the so-called pursuit-evasion problem[5]. The camera motions are usually controlled via an active vision system, and the motion can also be integrated into control servo loops [6][16].

Similar planning problems about object tracking were considered in [3] for robotics applications. A general dynamic programming approach was used to generate motions for a mobile robot (as an observer) to keep track of a moving target with a (partially) known trajectory. If the motion of the moving target is predictable, an optimal

tracking motion for the observer can be generated in an off-line manner. If the target's motion is only partially predictable, on-line motion strategies are used instead. The problem we consider in this paper belongs to the first case. However, the planning time of the off-line planner is too long (about 20 sec. on a modern workstation) to be used in our interactive application even for a simplified view model. Therefore, we have to look for a more efficient algorithm possibly by sacrificing global optimality.

1.2. Paper Organization

In the next section, we give a formal description of the tracking problem we consider. Several optimization criteria used in searching for a tracking motion are also described in details. In Section 3, we present a novel approach based on a best-first search algorithm aiming to find a feasible tracking motion as quickly as possible. Implementation details of this planner are given in Section 4 while experimental data and illustrative examples are reported in Section 5. We conclude the paper by discussing possible extensions of this planning system in the last section.

2. The Planning Problem

2.1. Problem Space Formulation

We assume that a moving *target t* and a *viewpoint v* (or *camera* interchangeably) exist in a bounded 2D Euclidean workspace *W* cluttered with obstacle regions *B*. The *configurations* of *t* and *v* are parameterized by $q^t=(x^t, y^t, \theta^t)$ and $q^v=(x^v, y^v, \theta^v)$, respectively. A configuration of an object is *collision-free* if and only if regions occupied by the object do not interfere with *B*. The *configuration space* (*C-space* for short) of *t* and *v*, denoted by C^t and C^v, respectively, are the spaces of all possible configurations. Let C^t_{free} and C^v_{free} denote the collision-free portions of their C-spaces, respectively. Then, the composite free C-space $X_{free} = C^t_{free} \times C^v_{free}$ is a 6D space, where our solution paths should reside. We suppose that the target's trajectory is given as a function of time and that all q^t's in the trajectory are collision-free. In other words, we will try to find a collision-free path in the configuration-time space of $CT(x^v, y^v, \theta^v, t) = (x^t(t), y^t(t), \theta^t(t), x^v, y^v, \theta^v)$, as formulated in [8][13]. However, not all configurations in *CT* are legal configurations since a legal configuration must also satisfy the visibility and velocity constraints of the camera. That is, in a legal configuration, the camera must be able to see portion of the target model with its view model (e.g. a view cone) as well as obey the maximal velocity constraint imposed on the camera.

To facilitate planning, transforming a configuration space to a different one might be preferred. In general, Cartesian parameters are more intuitive for specifying a viewpoint configuration; however, other parameterizations may be more adequate for formulating our planning problem. Since the viewpoint's goal is to track the target's motion, it makes sense to specify the viewpoint's configuration with respect to the target's coordinate system. For example, we can specify the viewpoint's relative configuration $q^{v'}$ with an alternative set of parameters (ϕ, l, φ) as shown in Figure 1. It is a one-to-one mapping between these two parameterizations. As we will show in the next section, such a relative parameterization will help us decouple the problem further to achieve on-line planning efficiency. Let $CT' = C^{v'} \times T$ be the new configuration-time space that we will consider in the rest of the paper.

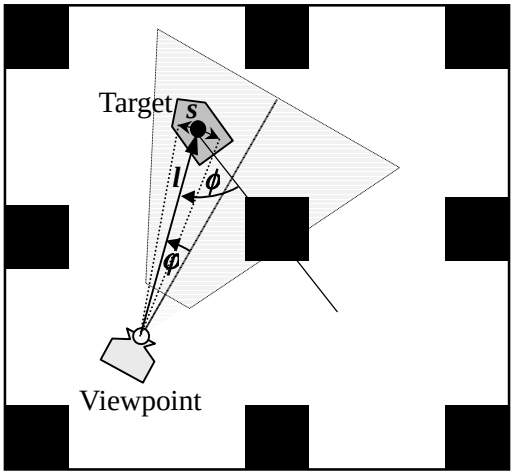


Figure 1: viewpoint parameters in a relative configuration

2.2. Search Criteria

When searching for a feasible path, we need to consider *soft* constraints as well as *hard* constraints. Hard constraints, such as configuration and visibility constraints, can be easily defined with geometric terms. For example, we define that the target is visible if and only if a fixed width centered on the target, denoted by *s* in Figure 1, can be seen by the camera all the time. However, the soft constraints, such as preferred viewing parameters, can only be defined according to user preferences and characteristics of the applications under consideration. The criteria that can be used to search for a good tracking motion are summarized as follows.

I. **Planning efficiency:** Since the planner will be used in an interactive manner to dynamically generate viewpoint motions, efficiency is the most important criterion to consider in our case. The planner should return the first feasible trajectory it finds that satisfies both the configuration and the visibility constraints. This criterion corresponds to the time dimension in CT'.

II. **Tracking direction:** When we simulate the motion of a camera following a tour guide, the position of the camera is usually preferably behind the tour guide. We define *tracking direction*, corresponding to the ϕ dimension in CT', as the orientation of the vector connecting the target and the viewpoint in the target's coordinate system. We force the tracking direction to stay within a range of orientations centered at the orientation right behind the target.

III. **View distance:** In order to maintain a clear image of

the tour guide in the field of view (FOV) of the camera, near and far clipping distances may need to be applied to further constrain the viewpoint motion. In fact, there should be a preferred view distance that is kept as closely as possible. This criterion corresponds to the l dimension in CT'.

IV. **Overall movement:** Although viewpoint movement in computer animation does not consume any energy as a human visitor does, the movement should still be minimized for the following reasons. Frequent viewpoint movements may introduce scene discrepancy and cause motion sickness. In addition, if the viewpoint movement is small, the underlying display module is more likely to take advantage of this characteristic to speed up 3D rendering. Therefore, it is desirable to reduce the movement, denoted by d, made in each step of the tracking trajectory as much as possible. The location of the viewpoint (x, y) is a function of ϕ and l. Therefore, d can be easily computed by comparing the current and the previous locations of the viewpoint.

V. **View angle:** View angle, the φ dimension in CT', is the angle between the direction that the camera is pointing to and the vector connecting the viewpoint and the target. The moving target may move out of sight if the view angle is outside the range of certain cutoff angles. It is desirable to keep the target clearly at the center of the view whenever possible but it also introduces undesirable side effects such as frequent scene changes as mentioned above. Hence, how to utilize this factor to achieve good viewing effects is a subjective matter that can only be defined by the user.

3. The Proposed Approach

In this section we propose a best-first planning algorithm to find a collision-free tracking motion in the relative configuration-time space, CT', defined in the previous section. The main difference between our approach and the dynamic programming approach described in [3] is on planning efficiency. However, as shown in many path planners[13], a four-dimensional space, such as CT', is usually too large to search for an interactive application. It might take several seconds to several minutes to find a path in the worst case. Therefore, the problem needs to be further simplified or decoupled. In order to reduce the dimensionality of our problem, we choose to decouple the φ-dimension from the rest of the space for the following reasons. First, we can always model the viewpoint as an enclosing circle such that any orientations of this circle will not violate the configuration constraint. Second, as long as the viewpoint is allowed to rotate as fast as the moving target, we can always adjust the view angle passively to maintain the visibility with the target. Therefore, by using a more conservative geometric model and a fixed view angle for the viewpoint, we can decouple the view angle φ from the rest of the configuration parameters. We will account for this φ-parameter after the other parameters of the tracking motion have been found.

```
procedure BFP {
  mark all the configurations in CT″ as unvisited;
  INSERT(q_i, OPEN); mark q_i as visited;
  SUCCESS ← false;
  while (!EMPTY(OPEN) and !SUCCESS) {
    q ← FIRST(OPEN);
    for (every q′ ∈ NEIGHBOR(q)) {
      mark q′ visited;
      if (LEGAL(q′)) {
        PARENT(q′) ← q;
        INSERT(q′, OPEN);
      }
      if (GOAL(q′)) SUCCESS ← true;
    }
  }
  if (SUCCESS)
    return the path by tracking back to q_i
}
```

Figure 2: Psudocode for the standard Best-First Planning (BFP) algorithm

With this simplification, we will focus on searching in the three-dimensional configuration-time space, CT'', comprised of configurations of three parameters: (t, ϕ, l).

We represent CT'' by a three-dimensional uniform grid as proposed in [9][13]. We assume that the viewpoint's velocity is upper bounded but its acceleration is not because no mass is associated with the virtual camera. This velocity bound will confine the search in the regions that are reachable from the initial configuration. We represent the search space by slices of 2D configuration spaces with each slice being incremented by Δt in the time-dimension. A configuration can only move forward in time to a neighboring configuration. The sizes of the grids are chosen such that moving from one configuration to one of its neighbors in the next time slice will not violate the maximal velocity limit.

Our planner uses a standard best-first planning (BFP) algorithm, as listed in Figure 2, to search for a collision-free path. The search starts from $q_i(t_s, \phi_i, l_i)$ and tries to find a path connecting this initial configuration to any legal goal configuration $q_g(t_e, *, *)$ in the last time slice ('*' represents any value). The algorithm returns the first feasible path if one exists and returns failure, otherwise. During the search process, a configuration is considered to be legal if and only if 1) all of its parameters are within legal bounds (including the velocity bound); 2) the viewpoint's geometric model does not collide with the obstacles; and 3) the view cone spanned by a fixed width on the target must not interfere with any occlusive obstacles.

In the BFP procedure, the FIRST operation returns the configuration of the lowest cost in the OPEN list to explore its neighboring configurations. In the interactive application under consideration, planning based on optimality of a specific criterion may not yield the most desirable motion as a whole. In our planner, the "best" condition is evaluated through a cost function f, in which the criteria

listed in the previous section are all taken into consideration. More specifically, we try to minimize the following cost function during the best-first search:

$$f(t, \phi, l, dir) = w_1 f_1(t) + w_2 f_2(\phi) + w_3 f_3(l) + w_4 f_4(\phi, l, dir) \quad (1)$$

$f_1(t) = t_e - t$
$f_2(\phi) = |\phi - \phi_0|$
$f_3(l) = |l - l_0|$
$f_4(\phi, l, dir) = dist(p(\phi, l, 0), p(\phi, l, dir))$,

where

w_i:weights for individual cost functions,
t: current time,
ϕ:current tracking direction,
l:current distance between the viewpoint and the target,
dir:an integer indicating the direction where the current configuration was created (0 means unchanged),
f_1: cost function for the distance between the current and the ending time slices. t_e is the ending time,
f_2: normalized cost function for the tracking direction. ϕ_0 is a preferred neutral tracking direction,
f_3: normalized cost function for the view distance. l_0 is a preferred neutral view distance,
f_4: normalized cost function for the Euclidean distance moved from the parent configuration,
p: returns the previous position of the viewpoint for the given approaching direction,
dist: returns the distance between two positions.

The overall cost function is a linear combination of the individual cost functions. The weights for these cost functions are subject to users' preferences. However, since the planning time is the main concern of our application, we use a heaviest w_1 for f_1 to make it a dominant cost function. Subjective viewing effects can be achieved by tuning the weights for each individual cost function at any time instance.

In the BFP procedure, the returned path consists of a sequence of configurations indexed by time. Two post-processing steps are applied to improve the found path. First, the path is run through a smoothing routine that tries to replace portions of the path with straight-line segments of the same lengths in the configuration-time space, provided that the accumulated costs for the new segments are smaller. Secondly, the view angle φ of the viewpoint, which was temporarily locked up during the search, is then computed. We unlock φ and allow it to change in a way that minimizes the viewpoint's rotational movement. That is, we do not rotate the viewpoint unless the target is going to exit the view cone.

4. Implementation

We have fully implemented the planning algorithm described in the previous sections. The planner, written in Java, reads in data files containing 2D geometric description of the projected objects in the workspace. At the current development stage, the planner contains two parts: a path planning module that computes sequences of holonomic motions for the target, and a motion tracking module that computes tracking motions for the viewpoint.

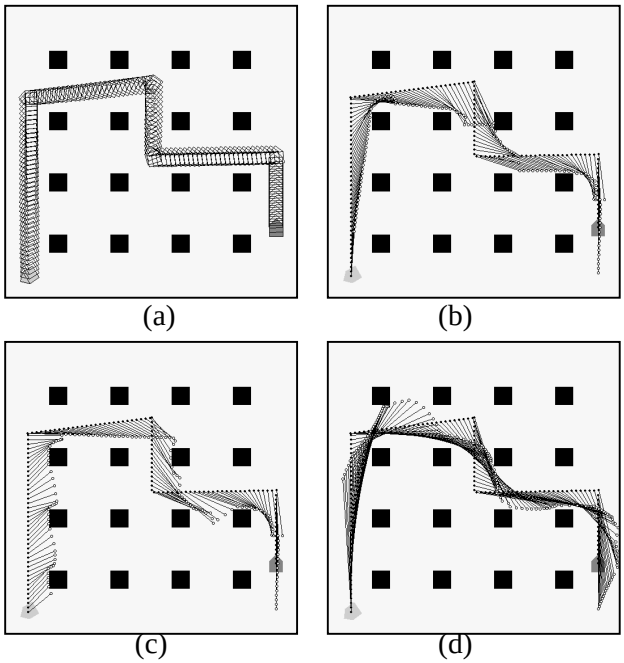
The search space for the tracking motion is an array of 2D configuration spaces, each of which corresponds to a specific time step. Since the planner takes a numerical approach, all motions found are accurate up to a certain resolution based on the Euclidean distance. Let the maximal transition increment be denoted by Δp. Then, the time difference between two slices, Δt, can be determined according to the given maximal velocity v_{max} ($\Delta t = v_{max} / \Delta p$). The number of time steps can then be computed accordingly, and the target's motion is resampled. The view distance is kept in a user-specified range centered at a neutral distance. The number of grid points along this dimension, n_l, is also computed according to Δp ($n_l = (l_{max} - l_{min}) / \Delta p$). The ϕ–dimension of the configuration space also spans a range of angles centered right behind the target. The increment, $\Delta\phi$, along this dimension is determined in a way such that the amount of movement caused by $\Delta\phi$ does not exceed Δp ($\Delta\phi = l_{max} / \Delta p$). The number of angle increments can then be determined.

In the BFP procedure, we maintain a priority queue, called OPEN, of configurations for further exploration. In each step of the process, the configuration of the lowest cost is chosen to be the next candidate to explore. As mentioned in the previous section, the dominant cost is the time difference between the current and the ending time slices, and the other costs are secondary costs that will be minimized whenever possible. Therefore, we use a two-level list to maintain this priority queue. The first level of list is a fixed-size array of references to a second level of linked lists of configurations. Each second-level linked list contains configurations of the same time index. The first nonempty linked list of the largest time index will always be the first to consider. We organize the linked list in a regular priority queue that supports INSERT and EXTRACT_MIN operations. The node of minimal cost computed according to eq(1) will then be extracted by the FIRST operation in the BFP procedure.

Only legal configurations can be inserted to the priority queue, and a legal configuration must be collision-free and satisfy the visibility constraint. In our current implementation, the viewpoint is assumed to be a small circle and its interference with the obstacles is checked on the fly. The visibility with the target is determined by checking whether the visibility cone collides with occlusive obstacles or not. The visibility cone is defined as the line segments originating from the viewpoint and spanning the must-see width on the target

5. Experiments

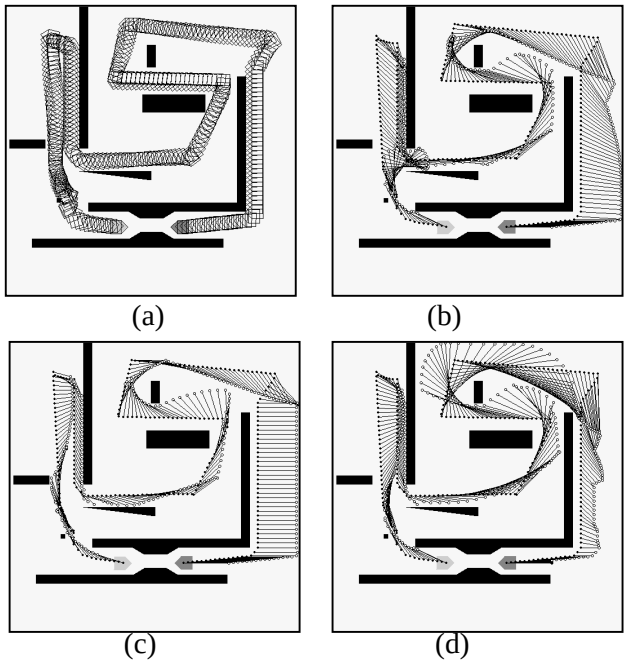
The system runs on regular desktop PC's or UNIX workstations and provides a graphical user interface for a user to specify the target's goals and personal tracking preferences such as the weights for various cost functions. In general, the running time of the algorithm is linear in the number of time steps. The actual running time mainly depends on the volume of the regions visited during the search process. In the worst case where no feasible paths exist, the total number of cells in the free configuration-



cases	w_2 (view distance, l)	w_3 (view angle, θ)	w_4 (movement, d)	planning time (seconds)
(b)	100.0	0.0	0.0	0.56
(c)	0.0	100.0	0.0	0.39
(d)	0.0	0.0	100.0	2.59

(e)

Figure 3: An example of viewpoint tracking motion with emphases on different criteria.



cases	w_2 (view distance, l)	w_3 (view angle, θ)	w_4 (movement, d)	planning time (seconds)
(b)	100.0	0.0	0.0	1.86
(c)	0.0	100.0	0.0	2.07
(d)	0.0	0.0	100.0	5.00

(e)

Figure 4: Another example of viewpoint tracking motion with emphases on different criteria.

time space can still be visited in a few seconds. In most cases, the running times (measured on a Pentium II 233 MHz PC) range from fractions of a second to a few seconds.

Figures 3-4 show two examples of workspaces cluttered with obstacles (configuration and occlusive obstacles are assumed to be the same). In both examples, we use the following parameter set: The workspace resolution is a uniform grid of 128x128 and each rotational increment is 3 degrees. The minimal, neutral, and maximal view distances (l_{min}, l_0, l_{max}) are 12, 20, and 60 units, respectively. The neutral view angle (φ) is set such that the target is at the center of the view, and it spans over a range of 30 degrees (±15 degrees). The tracking direction (ϕ) spans a range of 220 degrees (±110 degrees) with the neutral position being right behind the target. The target's path lengths (time steps) in the first and second examples are 257 and 515, respectively. For each example, we show the target's trajectory in sub-figure (a), the tracking motions with different optimization criteria in sub-figures (b)-(d), and weights and planning times for each case in sub-figure (e). Note that the target in both examples are moved sidewise to make tracking more interesting. In case (b), the viewpoint's motion emphasizes on maintaining a fixed view distance from the target. In case (c), the viewpoint tried to stay behind the target whenever possible. In case (d), the viewpoint tried to make a shortest move in each step whenever possible.

6. Conclusion and Future Work

Recently visibility planning has attracted great attentions in several robotics applications. Although the problem is similar to the basic path-planning problem, several additional factors to consider in interactive applications make it a new challenging task. In this paper, we have presented a novel approach that can provide a feasible tracking motion for the viewpoint in fractions of a second for interactive applications. This approach formulates the problem in a more intuitive space and adopts a best-first planning algorithm to search for a feasible tracking path. Since we emphasize on the response time of the planning system for on-line usability, the found path may not be optimal with respect to certain criteria. Nevertheless, for all of the examples created in our experiments, the planner produces satisfactory tracking motions that can be used in interactive applications.

We are in the process of integrating the planner into a virtual presence system that motivated this work. A snapshot of the example with the 2D traces of target and viewpoint is shown in Figure 5. The view model used in the planner will be verified and possibly adjusted for better usability. We also hope that we can automatically derive the viewing parameters used in the planner based on more intuitive user specifications.

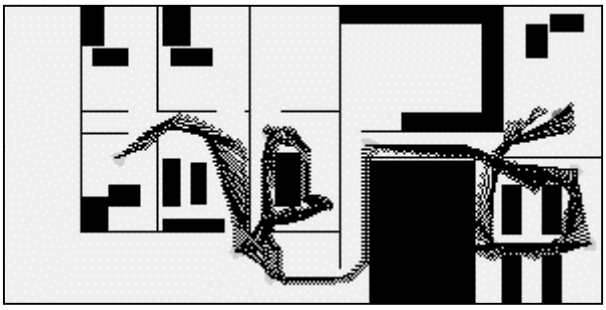


Figure 5: Snapshot of the planner running for a tour-guiding example inside a building.

Efficiency is the main concern of our planner since it is for an interactive application. Although most tracking motions can be found in fractions of a second and is quite satisfactory for our application, there are still rooms for improving efficiency and flexibility of the planner. In the current planner, configuration collisions and visibility occlusion are computed on the fly during the search. An alternative approach would be to systematically compute the forbidden regions in the configuration-time space in a preprocessing step and check for legality of a configuration in a lookup table during the search process. In more complex examples, the computation cost spent in this preprocessing step might save overall time in the long run. Examples of efficient algorithms to compute the configuration-time space can be found in [14][13]. Currently, generating a tracking motion is only a one-shot computation in our planner since we assume that the touring system will perfectly execute the tour plan. However, it is a desirable feature to allow a user to interactively modify the plan during the tour. The efficiency of the current planner should allow this to happen by calling the planner to locally or globally replan its motion to reflect the changes.

Acknowledgements

This work was partially supported by a grant from National Science Council under NSC 88-2218-E-004-002.

References

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

[2] A. J. Briggs and B. R. Donald. "Robust Geometric Algorithms for Sensor Planning," J.-P. Laumond and M. Overmars, editors, *Proceedings of 2nd Workshop on Algorithmic Foundations of Robotics*. A.K. Peters, Wellesley, pp. 197-212, MA, 1996.

[3] M. Gleicher and A. Witkin, "Through-the-Lens Camera Control," In E.E. Catmull, editor, *Computer Graphics (SIGGRAPH'92 Proceedings)*, volume 26, pp. 331-340, 1992.

[4] 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," *Proceedings of the Eighth International Symposium of Robotics Research*, Hayama, Japan, October 3-7, 1997.

[5] L. J. Guibas, J.-C. Latombe, S. M. LeValle, D. Lin, and R. Motwani, "Visibility-Based Pursuit-Evasion in a Polygonal Environment," *Proceedings of the 5th Workshop on Algorithms and Data Structures*, Springer Verlag, pp. 17-30, 1997.

[6] L-W He, M. F. Cohen and D. H. Salesin, "The Virtual Cinematographer: a Paradigm for Automatic Real-Time Camera Control and Directing," *Proceedings of ACM SIGGRAPH'96*, pp. 217-224, 1996.

[7] S. Hutchinson, G.D. Hager, and P.I. Corke, "A Tutorial on Visual Servo Control," *IEEE Transaction on Robotics and Automation*, 12(5):651-670, October 1996.

[8] K. Kant and S. W. Zucker, "Toward Efficient Trajectory Planning: The Path-Velocity Decomposition," *International Journal of Robotics Research*, 5(3):72-89, 1986.

[9] J.-C. Latombe, "*Robot Motion Planning*," Kluwer Academic Publisher, Boston, MA, 1991.

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

[11] 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.

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

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

[14] T. Lozano-Perez, "Spatial Planning: A Configuration Space Approach," *IEEE Transaction on Computers*, 32(3):108-120, 1983.

[15] J. O'Rourke, "*Art Gallery Theorems and Algorithms*," Oxford University Press, NY, 1987.

[16] N. P. Papanikolopoulos, P. K. Khosla and T. Kanade, "Visual Tracking of a Moving Target by a Camera Mounted on a Robot: A Combination of Control and Vision," *IEEE Transaction on Robotics and Automation*, 9(1):14-35, February 1993.

[17] S. Teller, and P. Hanrahan, "Global Visibility Algorithms for Illumination Computations," *Proceedings of SIGGRAPH'97*, pp. 239-246, 1997.

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