Path Planning for Complex Terrain Navigation Via Dynamic Programming*

Kwan S. Kwok
Sandia National Laboratories, Albuquerque, NM 87185-1003, kskwok@sandia.gov
Brian J. Driessen
Sandia National Laboratories, Albuquerque, NM 87185-0439, bjdries@sandia.gov

Abstract. This work considers the problem of planning optimal paths for a mobile robot traversing complex terrain. In addition to the existing obstacles, locations in the terrain where the slope is too steep for the mobile robot to navigate safely without tipping over become mathematically equivalent to extra obstacles. To solve the optimal path problem, we use a dynamic programming approach. The dynamic programming approach utilized herein does not suffer the difficulties associated with spurious local minima that the artificial potential field approaches do. In fact, a globally optimal solution is guaranteed to be found if a feasible solution exists. The method is demonstrated on several complex examples including very complex terrains.

1. Introduction

Path planning for mobile robots has involved all of the extremes in terms of available environment information. The worst case is that in which the robot's collision-avoidance mechanism is simply to "repel" from obstacles that it comes very close to [Brooks, Arkin]. Sensors involved include proximity sensors. An intermediate case is one in which the robot has a camera or sonar [Cho and Lim 1, Cho and Lim 2] and can "see ahead" a limited distance, much like a human would when driving an ATV though rough and/or obstacle-infested terrain. The best case is that in which knowledge of all obstacles is available so that the vehicle can be guaranteed to find an optimal path if a feasible path exists (see [Hou and Zheng] and the references cited therein). It is this latter best-case scenario that is of concern in this paper.

All such problems can be reduced to that of finding an optimal path through a sequence of adjacent feasible or admissible locations, where the inadmissible or "forbidden" locations are obstacle locations. This work will focus on the problem of complex terrain navigation in which forbidden regions arise from not only obstacles such as buildings, fences, and bodies of water but also from excessively steep terrain that could cause the vehicle to tip over. This work will focus upon planning minimum-distance paths (which are essentially minimum-time paths) where it is reasonable to expect that travel-time has some correlation with probability of being detected/destroyed by the enemy. The method used is a dynamic programming [Larson and Casti] one which is guaranteed to produce a globally optimal solution. This globalness of solution contrasts with previous work that use artificial repulsive potentials for obstacles (see [Hou and Zheng] and the references cited therein) because these latter approaches can lead to not only sub-optimal paths but can also completely fail to find a feasible path, even when one exists, both of these cases being related to the problem of spurious local minima. Despite these difficulties with the potential approach, it is of interest because it can be computationally cheaper; however we have found the cost of solving for globally optimal solutions to be trivial for the problems considered herein so that the computational consideration was not an issue.

2. Problem Statement

The problem considered herein is that of finding a minimum-distance, and hence

* Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000

minimum-time, feasible path over complex terrain between a given starting position and given goal position. Many locations on the terrain are "forbidden" because the slope of the terrain is too steep for the mobile robot to navigate safely without tipping over. There exist other forbidden regions of the domain that act as additional obstacles. These may include bodies of water or regions where the probability of detection by the enemy is known to be high. Other cost functions, besides distance traveled, could also be considered. For example, the probability of being detected by the enemy could be used as a cost function, where each location in the terrain has an associated probability of detection. The cost function would then be multiplicative instead of additive but can still be handled by the dynamic programming approach utilized herein.

3. Method of Solution

The domain with hills defined by $z = z(x, y)$ and other obstacles will be gridded up into an array of equally-spaced grid points. Grid points are defined as forbidden if the grade (slope) of the terrain $z = z(x, y)$ is too large or if some other obstacle exists at that grid point. The rectangular array of grid points is enumerated the same way as the associated matrix. Let M denote such a matrix and let

$$M_{ij} = 0 \text{ if } (i,j) \text{ is forbidden} \qquad (3.1)$$

$$M_{ij} = 1 \text{ if } (i,j) \text{ is not forbidden} \qquad (3.2)$$

Let tensor C_{kij} denote the "cost to go" for the kth time step back from the final time at the (i,j) grid location. In one time step, the vehicle can make one of the following 8 moves which constitute the set of all possible diagonal and non-diagonal moves

$$(i \leftarrow i+1, j \leftarrow j-1),\quad (i \leftarrow i+1, j \leftarrow j),$$
$$(i \leftarrow i+1, j \leftarrow j+1),\quad (i \leftarrow i-1, j \leftarrow j-1),$$
$$(i \leftarrow i-1, j \leftarrow j),\quad (i \leftarrow i-1, j \leftarrow j+1),$$
$$(i \leftarrow i, j \leftarrow j-1),\ (i \leftarrow i, j \leftarrow j+1) \qquad (3.3)$$

If we let (i_{end}, j_{end}) denote the final required destination, then clearly the "cost to go" for k=1 is given by

$$C_{1ij} = \begin{Bmatrix} -1, \text{ if } \mathrm{i} \neq \mathrm{i}_{\mathrm{end}} \text{ or } \mathrm{j} \neq \mathrm{j}_{\mathrm{end}} \\ 0, \text{ otherwise} \end{Bmatrix} \qquad (3.4)$$

where a $C_{kij} = -1$ will be used to indicate that location (i,j) is not a valid location for the kth backward time step. Equation (3.4) simply states that the only valid location at the last time step is (i_{end}, j_{end}), i.e., the vehicle must reach its destination.

Let $L_1(\hat{i}, \hat{j}, m)$ be the i value the vehicle moves to from $(\hat{i}, \hat{j})$ using the mth (of 8) move type (of those indicated in (3.3)). Let $L_2(\hat{i}, \hat{j}, m)$ be the j value the vehicle moves to from $(\hat{i}, \hat{j})$ using the mth (of 8) move type. Let $(i_{\max}, j_{\max})$ denote the grid point in the lower-right-most position of Figure 1. Let the function F be defined as follows:

$$F(i,j,m,i_{\max},j_{\max}) = \begin{Bmatrix} -1 \text{ if } \mathrm{L}_1(i,j,m) \notin (1,\ldots,i_{\max}) \text{ or} \\ \mathrm{L}_2(i,j,m) \notin (1,\ldots,j_{\max}) \text{ or} \\ M(\mathrm{L}_1(i,j,m), \mathrm{L}_2(i,j,m)) = 0 \\ \begin{pmatrix} \text{distance between (i, j)} \\ \text{and } (\mathrm{L}_1(i,j,m), \mathrm{L}_2(i,j,m)) \end{pmatrix}, \\ \text{otherwise} \end{Bmatrix} \qquad (3.5)$$

Let S_{kij} denote the following set of integers m:

$$S_{kij} = \begin{Bmatrix} m\colon\ m \in (1,\ldots,8), \\ C_{k,L_1(i,j,m),L_2(i,j,m)} \neq -1, \\ F(i,j,m,i_{\max},j_{\max}) \neq -1 \end{Bmatrix} \qquad (3.6)$$

Let $I_{k\hat{i}\hat{j}}$ denote the optimal i location to move forward-in-time to from the position $(\hat{i}, \hat{j})$ at backward time step k, and let $J_{k\hat{i}\hat{j}}$ denote the optimal j location to move to from the position $(\hat{i}, \hat{j})$ at backward time step k. Then the $C_{k+1,i,j}$, $I_{k+1,i,j}$, and $J_{k+1,i,j}$ $(\forall i, j)$ can be calculated from the C_{kij} $(\forall i, j)$ as follows:

If S_{kij} is an empty set, $C_{k+1,i,j} = -1$ (3.7)

Else

$$C_{k+1,i,j} = \min_{m \in S_{kij}} \left(F(i,j,m,i_{\max},j_{\max}) + C_{k,L_1(i,j,m),L_2(i,j,m)} \right) \qquad (3.8)$$

$$m^*_{k+1,i,j} \equiv \arg\min_{m \in S_{kij}} \left(F(i,j,m,i_{\max},j_{\max}) + C_{k,L_1(i,j,m),L_2(i,j,m)} \right) \qquad (3.9)$$

$$I_{k+1,i,j} = L_1(i,j,m^*_{k+1,i,j}) \qquad (3.10)$$

$$J_{k+1,i,j} = L_2(i,j,m^*_{k+1,i,j}) \qquad (3.11)$$

Endif

Let (i_{start}, j_{start}) denote the starting location of the vehicle. We proceed in this manner for k=1,2,3,4,... until one of two things happens:

Case (1): $C_{k+1,i_{start},j_{start}} \neq -1$ (3.12)

Case (2): The set of $C_{k+1,i,j}$ values for which $C_{k+1,i,j} \neq -1$ has the same number of members as the set of C_{kij} values for which $C_{kij} \neq -1$. (3.13)

In Case (1), an optimal solution has been obtained and is generated by

```
$i^* = i_{start}$                      (3.14)
$j^* = j_{start}$                      (3.15)
Loop $p$=1 to $k$-1
    $i^* = I_{pi^*j^*}$                (3.16)
    $j^* = J_{pi^*j^*}$                (3.17)
End Loop
```

thus giving us an optimal sequence of k grid positions that move us from (i_{start}, j_{start}) to (i_{end}, j_{end}).

In Case (2), no feasible solution exists as moving back one time step has not increased the size of the set of valid grid locations. This means that no matter how many more time steps we step back, the set will remain the same so that it will clearly never contain (i_{start}, j_{start}).

We should note that the cost to find the optimal sequence of moves is $O(\bar{n}N)$ where $\bar{n}$ is the number of grid points in the rectangular region and N the number of time steps (where N is not known a priori).

4. Numerical Examples

Figure 1 below illustrates an optimal path. The solid dots denote grid points that are not forbidden. All other grid points are forbidden. The circle denotes the starting position and the "+" denotes the goal position.

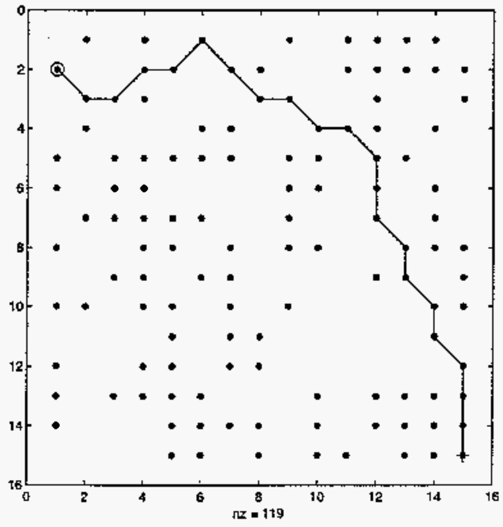

Figure 1. Example Time-Optimal Solution, nz Denotes the Number of Admissible Locations or "Stepping Stones"

Figure 2 below illustrates a problem that is infeasible. The algorithm of Section 3 recognizes the problem as infeasible at k=6 (the 6th backward time step).

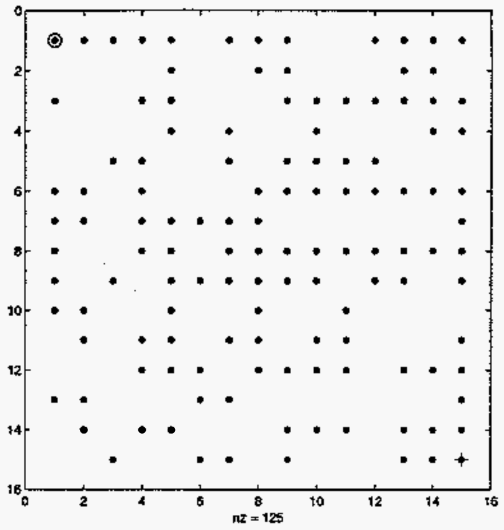

Figure 2. Example of Infeasible Problem, Automatically Recognized as Infeasible by the Algorithm

Figure 3 illustrates an optimal path for another case.

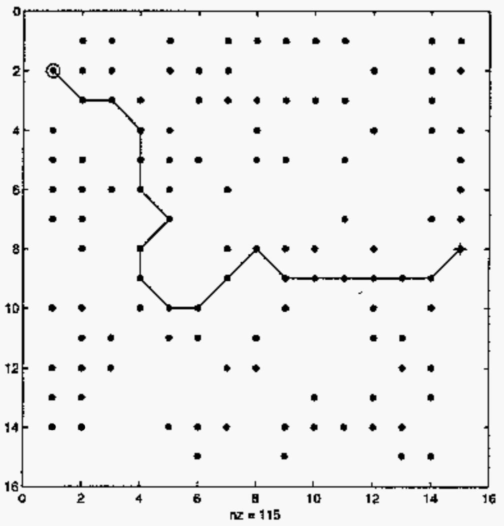

Figure 3. Example Time-Optimal Solution

We consider next a case in which the forbidden points are generated by an actual surface which is illustrated in Figure 4 below.

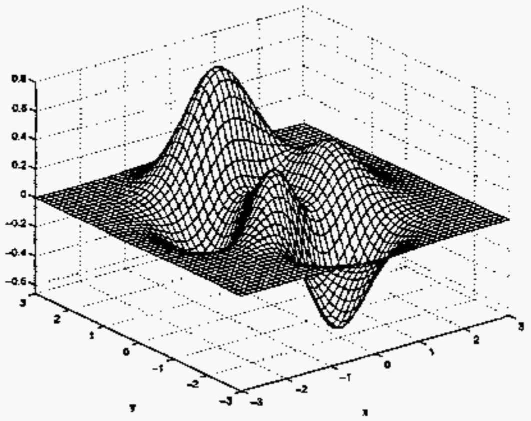
Figure 4. Illustration of a Surface Terrain

The allowable grade was set to 25 degrees. The starting point and goal point are illustrated in Figure 5 below.

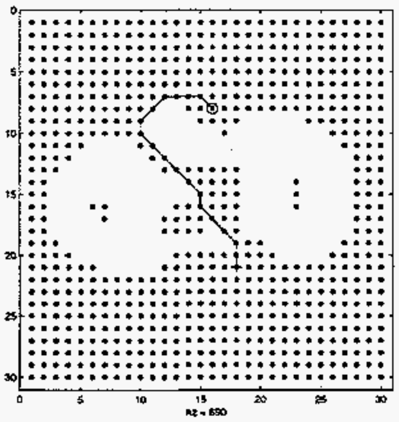
Figure 5. Illustration of Time-Optimal Path

In Figure 5 the upper left corner corresponds to the point $(x, y) = (-3, -3)$ in Figure 4. We see then from Figure 5 that the robot's net travel is mostly in the $+x$ direction.

Finally Figure 6 and Figure 7 below show how the solution changes as the grid is made coarser and coarser.

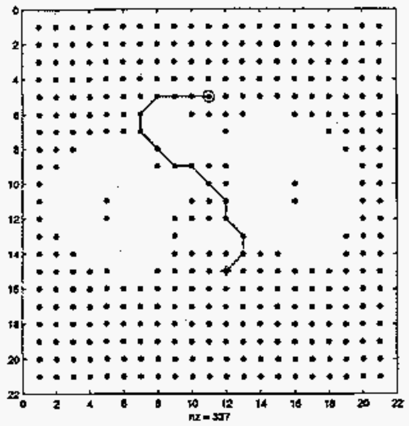
Figure 6. Illustration of Time-Optimal Path

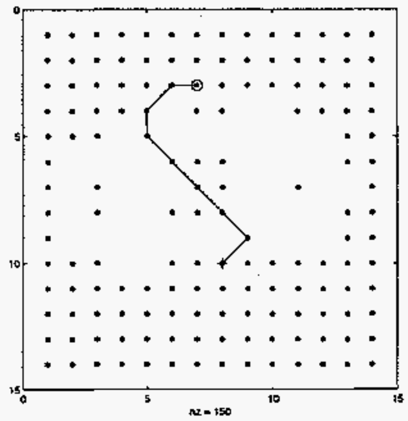
Figure 7. Illustration of Time-Optimal Path

5. Conclusion

This work demonstrated that dynamic programming could be used to obtain optimal paths for a mobile robot traversing complex terrain. Locations in the terrain at which the slope is too steep for the robot to navigate safely without tipping over become mathematically equivalent to obstacle locations, which are added to the set of existing obstacle locations. With only two state variables (x and y position), dynamic programming is very effective and is guaranteed to find a globally optimal path.

References

[1] Arkin, Ronald C., "Cooperation Without Communication: Multiagent Schema-Based Robot Navigation," *Journal of Robotic Systems* 9(3), 1992, pp. 351-364.

[2] Brooks, Rodney A. and Flynn, Anita M., "Fast, Cheap and Out of Control: A Robot Invasion of the Solar System," *Journal of*

the British Interplanetary Society, Vol. 42, 1989, pp. 478-485.

[3] Brooks, Rodney A., "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, March 1986, pp. 14-23.

[4] Hou, Edwin S. and Zheng, Dan, "Mobile Robot Path Planning Based on Hierarchical Hexagonal Decomposition and Artificial Potential Fields," *Journal of Robotic Systems*, Vol. 11, No. 7, 1994, pp. 605-614.

[5] Larson, R.E. and Casti, J.L., *Principles of Dynamic Programming, Part II, Advanced Theory and Applications*. New York: Marcel Dekker, Inc., 1982.